10 GIT ANTI PATTERNS
YOU SHOULD BE AWARE OF

Lemi Orhan Ergin
Agile Software Craftsman & Co-Founder, Craftbase
LEMi ORHAN ERGiN
agile software craftsman
co-founder @ craftbase
ex Sony, eBay, ACM, iyzico
founder & co-organizer of SCTurkey
Turkish Software Craftsmanship Community

/lemiorhan
lemiorhanergerin.com
@lemiorhan
Do you push every commit just after you create?
Do you push every commit just after you create it?
if all you do is commit and immediate push

you lose the opportunity to organize your commits via reset, rebase, commit with amend safely

and you have to use force push at every time you organize your commits
are you brave enough to jump to any commit?
are you brave enough to jump to any commit?

Nope. Usually tests do not pass, the application does not work, even the code does not compile in majority of commits.
are you brave enough to jump to any commit?

BROKEN TIME MACHINE ANTI-PATTERN

Do you want me to help you about finding the last commit that finalizes the feature?
do you have looooooong living topic branches?
do you have loooooong living topic branches?
LONG LIVING BRANCHES ANTI-PATTERN

do you have looooonng living topic branches?

welcome to 
merge hell

LONG LIVING BRANCHES ANTI-PATTERN
do you use **cherry-pick** to prepare releases?
do you use cherry-pick to prepare releases?
$ git cherry-pick Every-Single-Commit-We-Want-To-Deploy

CHERRY-PICK OVERDOSE ANTI-PATTERN
do you fully understand when you read the commit graph?
do you fully understand when you read the commit graph?
Do you fully understand the commit graph?

When you read the commit graph, do you fully understand topic and shared branches, tracking branches, tags, HEADs, merge commits, reverted commits...
Cure?
Commit Early, Commit Often Perfect Later, Publish Once

PS: There are more than one way to achieve this
Our very first step is defining a strategy to keep branches short.

split your big feature into mini shippable tasks

refactorings
tasks, like rest endpoints

red-green-refactor

each task will have a branch, not a feature
commit early
commit often
no need to compile
no need for CI
it’s only for versioning

do not push
Sync source branch with one simple command

$ git fetch origin master:master

STEP 2
Sync source branch with one simple command

$ git fetch origin master:master

STEP 2
Get incoming change sets from source to topic branch via merge

$ git merge master

STEP 3
Get incoming change sets from source to topic branch via merge

$ git merge master
$ git merge master

REGULARLY
Get incoming change sets from source to topic branch via merge

STEP 3
REGULARLY
Get incoming change sets from source to topic branch via merge

$ git merge master
merge back to source by squashing all commits in topic branch

$ git checkout master
$ git merge --squash topic
Delete the topic branch

$ git branch -D topic
Delete the topic branch

$ git branch -D topic
Delete the topic branch

$ git branch -D topic

Now you can push
Continuous Integration validates master branch continuously.

Pull requests can be used to review code and to validate before merging back to master.

Deleting branches after merge will make your commit graph readable.

Scrum tasks are mapped to commits, not stories.

TRUNK-BASED DEVELOPMENT
let's make git king again

Commit early & often, perfect later, publish once philosophy.

Deliver frequently, be prepared to send every single commit.

Github Flow can be used to govern overall philosophy.

Feature flags should be used whenever possible.
is it hard to learn Git commands?
use terminal

GUls are prison balls of developers

it’s ok to use GUls while checking diffs, resolving conflicts and viewing commit graph
does each of your commit have one special purpose?

or you commit anything you have changed
stop adding
every change
prevent commits from being big ball of muds
stop adding every change
prevent commits from being big ball of muds

local change sets at JetBrains IDEs
stop adding every change

partial add: `git add -p`
do you really understand what's written in commit messages?

* f7c82af - (HEAD -> feature/message-system, origin/feature/message-system) ini dosyaları
* 5bace68a - bug fixes (20 hours ago) <fmatak>
* 9688663 - Added MimeMessageFacade for inline emails (20 hours ago) <fmatak>
* 011f316 - Removed most of the classicalmailsender references (20 hours ago) <fmatak>
* 56b6e56 - Added immediate email and made refactorings (20 hours ago) <fmatak>
* 07bce61 - Added reservation email templates (20 hours ago) <fmatak>
* f3e8e91 - bug fix (20 hours ago) <fmatak>
* f337760 - Added detailed auditing and receiver address types (20 hours ago) <fmatak>
* 4f7da2c - Messaging after some refactoring 2 (26 hours ago) <fmatak>
* 0c166d2 - Messaging after some refactoring (26 hours ago) <fmatak>
* cc47338 - Messaging sent second email using new system (26 hours ago) <fmatak>
* ba1bf16 - Messaging sent first email using new system (26 hours ago) <fmatak>
* 0b3acb1 - Updated messaging and job architecture (26 hours ago) <fmatak>
* 6846a47 - db ye kayıt yapan hali (26 hours ago) <fmatak>
* a604198 - some refactoring (26 hours ago) <fmatak>
* c75d41b - Created DB tables and hibernate beans (26 hours ago) <fmatak>
* 4dc66d2 - pom.xml' deki common-utils' den jcl-over-slf4j dependency' si exclude edildi (26 hours ago) <fmatak>
* 239fc9f - Trying get tests working... (26 hours ago) <fmatak>
* 65abd42 - Spring ile JUnit test ornegi eklendi (SampleTest.java), spring-test dependency eklendi (26 hours ago) <fmatak>
* 90ec1c8 - Added Message Persister Interface (26 hours ago) <fmatak>
* b176749 - Messaging system project created, email sending refactored out, new interface (4 weeks ago) <fmatak>
* fc2e873 - Added rebelxml to the ignore list (4 weeks ago) <fmatak>
commit messages are documents!

use git commit templates to create better commit messages

$ git config --global commit.template ~/.git-commit-template.txt
$ git config --global commit.cleanup strip

# WHAT
# <issue id> (this commit will...) <subject>

# WHY and HOW
# Explain why this change is being made

# RELATED
# Provide links or keys to any relevant issues or other resources

# REMEMBER
# use lower case in the subject line
# start with a verb in imperative tone in the subject line
# do not end the subject line with a period
# separate subject from body with a blank line
# use the body to explain what and why vs. how
# can use multiple lines with "-" for bullet points in body
Which of the following commands can cause **duplicate commits** in the commit graph?

A  git rebase
B  git push --force
C  git merge
D  git pull
Which of the following commands can cause duplicate commits in the commit graph?

A. `git rebase`
B. `git push --force`
C. `git merge`
D. `git pull`
$ git rebase master
$ git push -f
REBASE & FORCE PUSH

$ git rebase master
$ git push -f
REBASE & FORCE PUSH

```
$ git rebase master
$ git push -f
```
$ git pull
PULL A FORCE-PUSHED BRANCH

$ git pull
PULL A FORCE-PUSHED BRANCH

$ git pull
PULL WITH REBASE

$ git pull --rebase
PULL WITH REBASE

$ git pull --rebase
1. TORTURING GIT BY PUSH
2. BROKEN TIME MACHINE
3. LONG LIVING BRANCHES
4. TOO LATE TO VALIDATE
5. CHERRY-PICK OVERDOSE
6. LOST IN COMMIT GRAPH
7. BUTTON ADDICT
8. TRASH HOUSE
9. BIG BALL OF MUD
10. AMBIGIOUS COMMIT MESSAGES
11. ZOMBIE REBASE
12. CODE LOSING SYNDROME
13. MESS UP WITH THE ROLLBACK
14. CENTRALIZED GIT
15. MERGE FANATIC
16. BRANCH CEMETERY
17. UNCONTROLLED POWER
18. WEB OF REPOSITORIES
19. ORACLE SYNDROME
20. WAITING FOR HACKERS
21. EVIL MERGE
22. BRANCH OVERDOSE
23. CHUCKY THE COMMAND
24. NO HERO TO SAVE LIVES
25. DUPLICATE COMMITS
26. BIG FAT COMMIT
27. CONFLICT-FOBIA
28. MERGE HELL
29. F*UCK UP WITH FORCE PUSH
30. LIVING AT DETACHED HEAD STATE
Let's recap what was really happened at that time?
$ git checkout tags/v1.1
Note: checking out 'cecd95914'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.
my poor friend worked for 3 long days
$ git checkout master

when he moved to another branch, all commits were gone
$ git reflog

aa67e3a2c HEAD@{0}: rebase finished: returning to refs/heads/fix/java-sql-Date-violates-LSR
aa67e3a2c HEAD@{1}: rebase: fixes UnsupportedOperationException while calling toInstant() method of java.sql.Date
a45f3c4e5 HEAD@{2}: rebase: checkout develop
630ddad6e HEAD@{3}: checkout: moving from develop to fix/java-sql-Date-violates-LSR
b26cf7a1a HEAD@{4}: rebase: checkout develop
630ddad6e HEAD@{5}: checkout: moving from develop to fix/java-sql-Date-violates-LSR
b26cf7a1a HEAD@{6}: pull: Fast-forward
8b59f8f50 HEAD@{7}: checkout: moving from fix/java-sql-Date-violates-LSR to develop
630ddad6e HEAD@{8}: rebase: updating HEAD
$ git reflog

aa67e3a2c HEAD@0: rebase finished: returning to refs/heads/fix/java-sql-Date-violates-LSR
aa67e3a2c HEAD@1: rebase: fixes UnsupportedOperationException while calling toInstant() method of java.sql.Date
a45f3c4e5 HEAD@2: rebase: checkout develop
630ddad6e HEAD@3: checkout: moving from develop to fix/java-sql-Date-violates-LSR
b26cf7a1a HEAD@4: rebase: checkout develop
630ddad6e HEAD@5: checkout: moving from develop to fix/java-sql-Date-violates-LSR
b26cf7a1a HEAD@6: pull: Fast-forward
8b59f8f50 HEAD@7: checkout: moving from fix/java-sql-Date-violates-LSR to develop
630ddad6e HEAD@8: rebase: updating HEAD
$ git branch typofix 630ddad6e
$ git branch typoﬁx 630ddad6e
thank you all!

Feedback: bit.ly/lemiorhan