

# INTRODUCTION TO GIT

IHAR PRYSHCHEPAU

OCTOBER, 2019



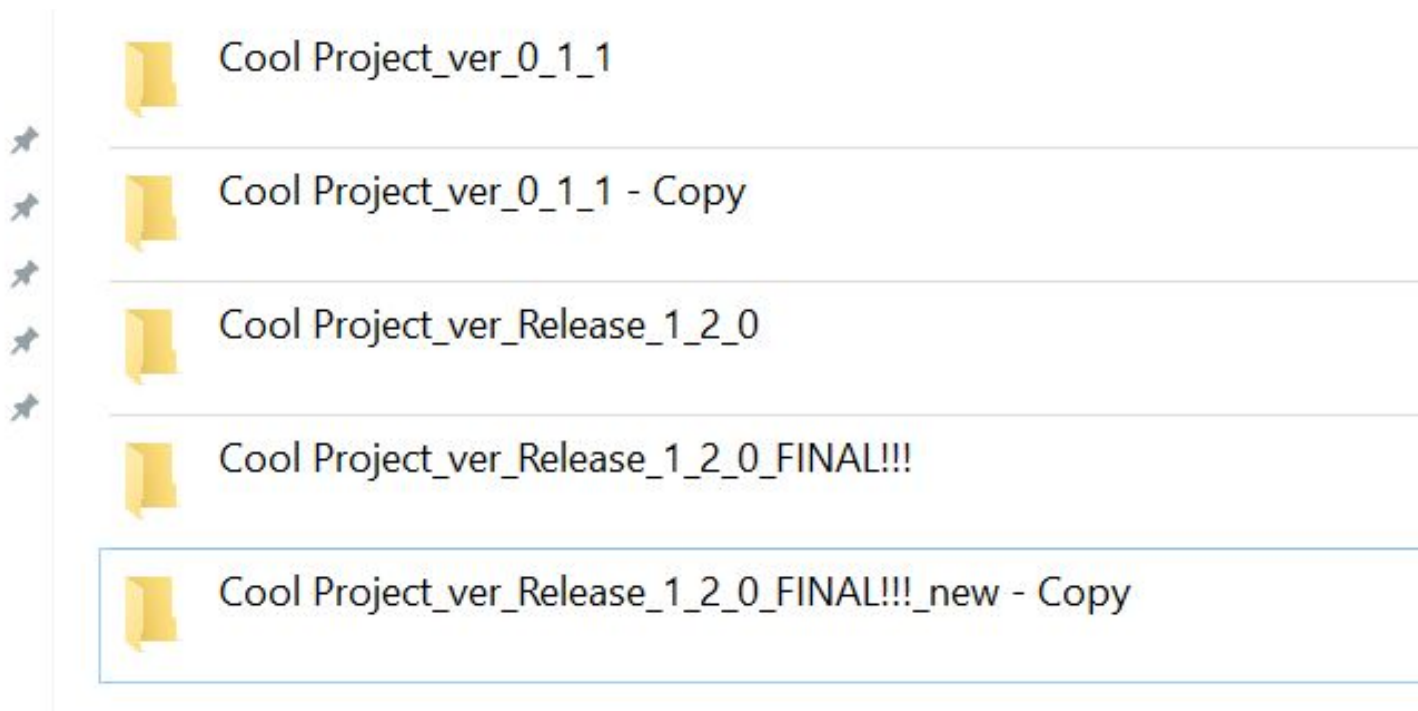
**iTechArt**

- ❑ Система контроля версий (VCS)
- ❑ Git

# УРОВЕНЬ 1. АВТОНОМНАЯ РАБОТА

3

**BAD PRACTICE**



# УРОВЕНЬ 2. NETWORK SHARING

4

**BAD PRACTICE!**

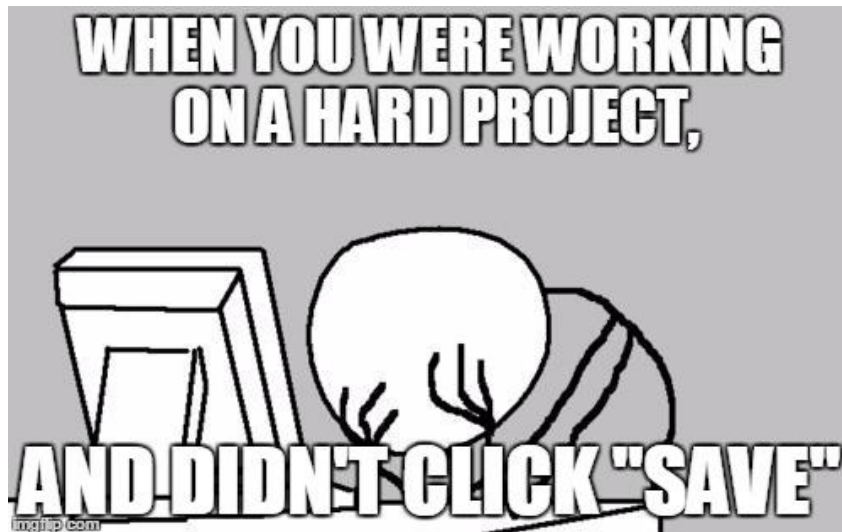
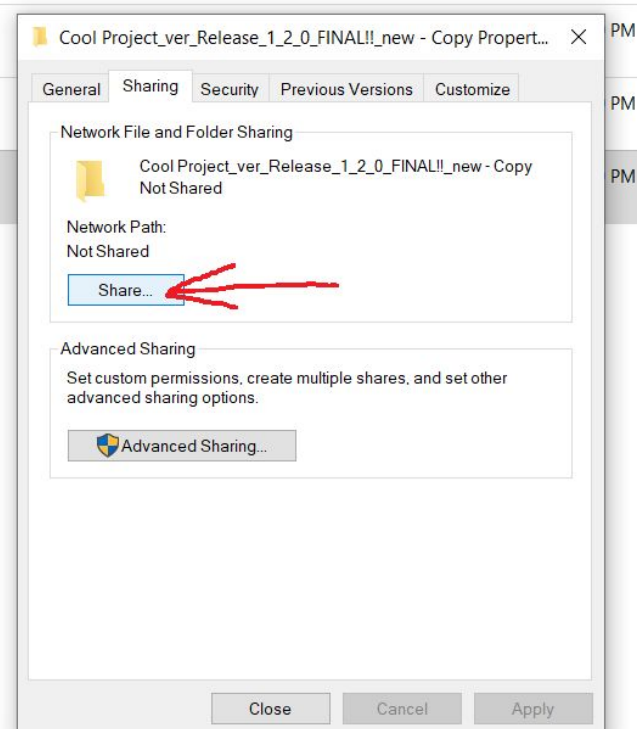
Cool Project\_ver\_0\_1\_1 - Copy

Cool Project\_ver\_Release\_1\_2\_0

Cool Project\_ver\_Release\_1\_2\_0\_FINAL!!!

Cool Project\_ver\_Release\_1\_2\_0\_FINAL!!\_new - Copy

Date modified: 9/9/2019 2:20 PM

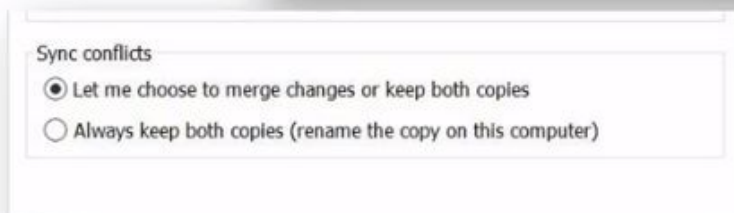
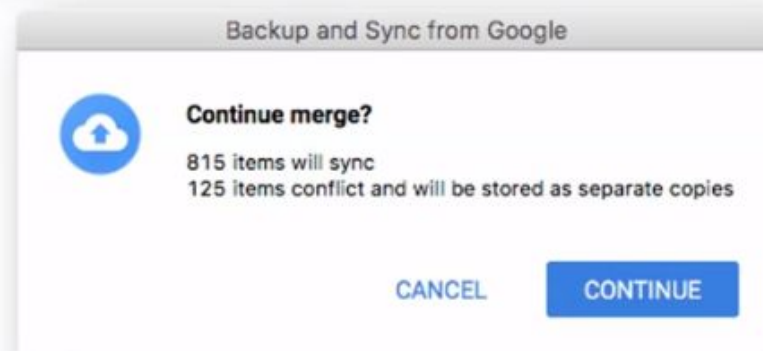


# УРОВЕНЬ 3. CLOUD

5

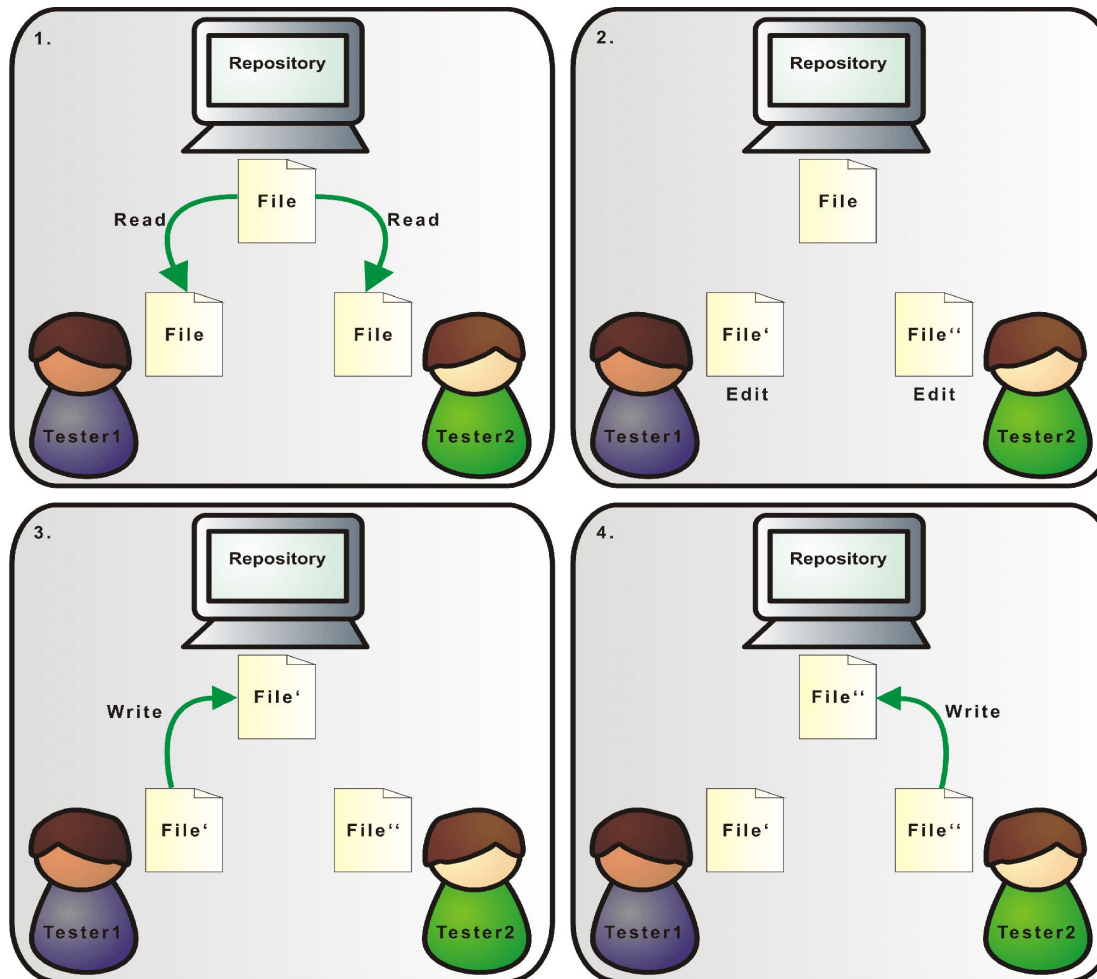


Name	Date Modified
Sample File	Today at 10:54
Sample File (hanz-mbp's conflicted copy 2018-10-12)	Today at 10:54



# ГЛАВНАЯ ПРОБЛЕМА КОНТРОЛЯ ВЕРСИЙ

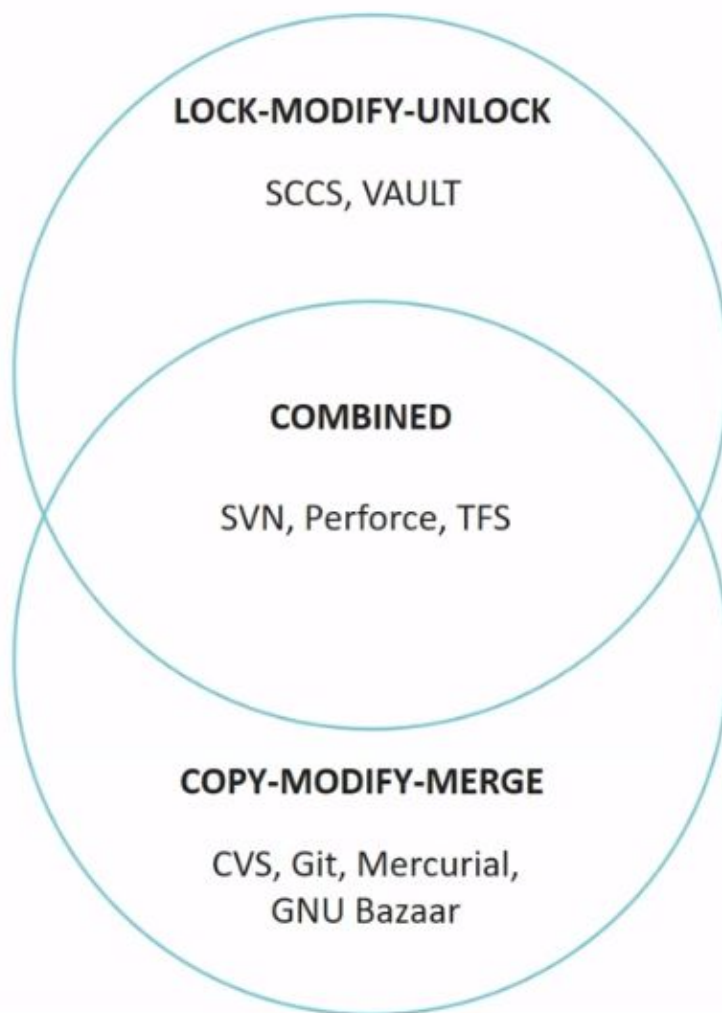
6



- ✓ Back-up & restore
- ✓ Синхронизация командной работы
- ✓ Отмена изменений
- ✓ Отслеживание изменений и их авторов
- ✓ Sandboxing
- ✓ Возможность работы с ветками

# ВИДЫ **VCS**: ПО МЕХАНИЗМУ РАБОТЫ С ФАЙЛАМИ

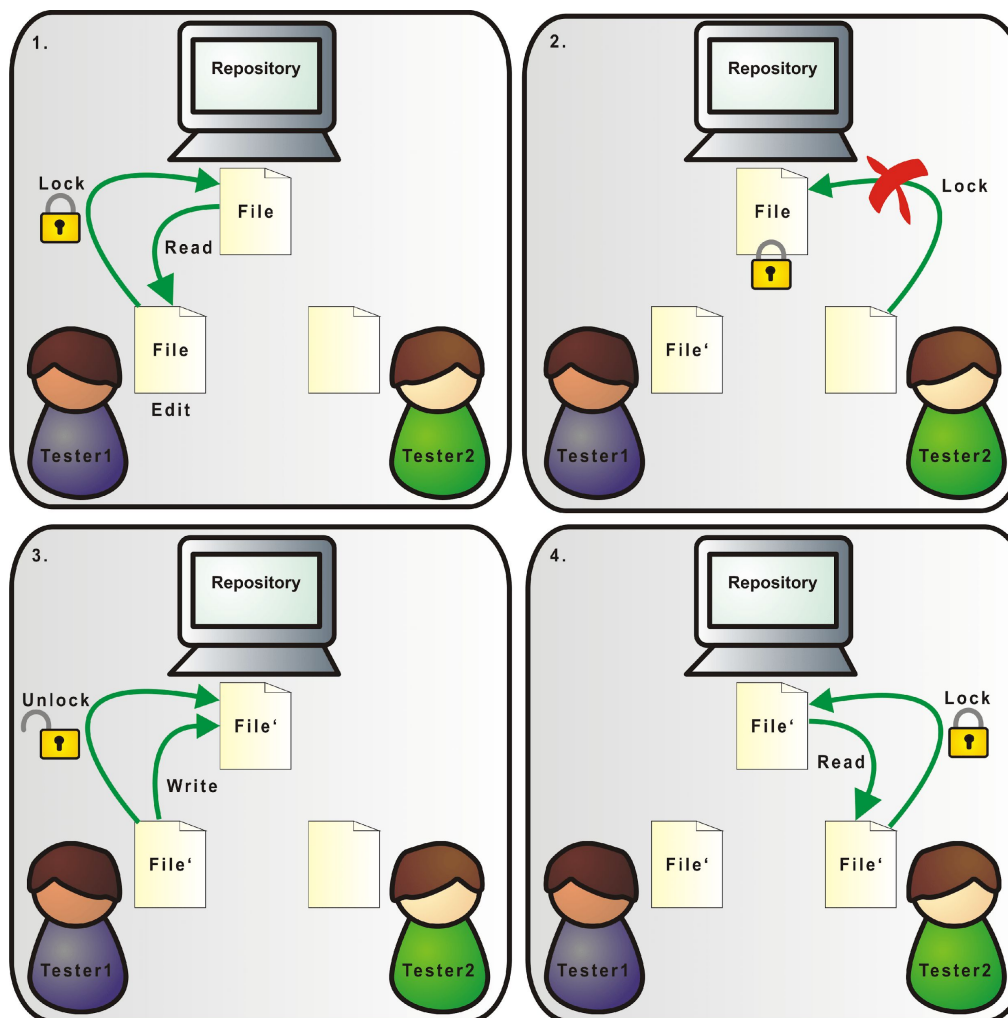
8





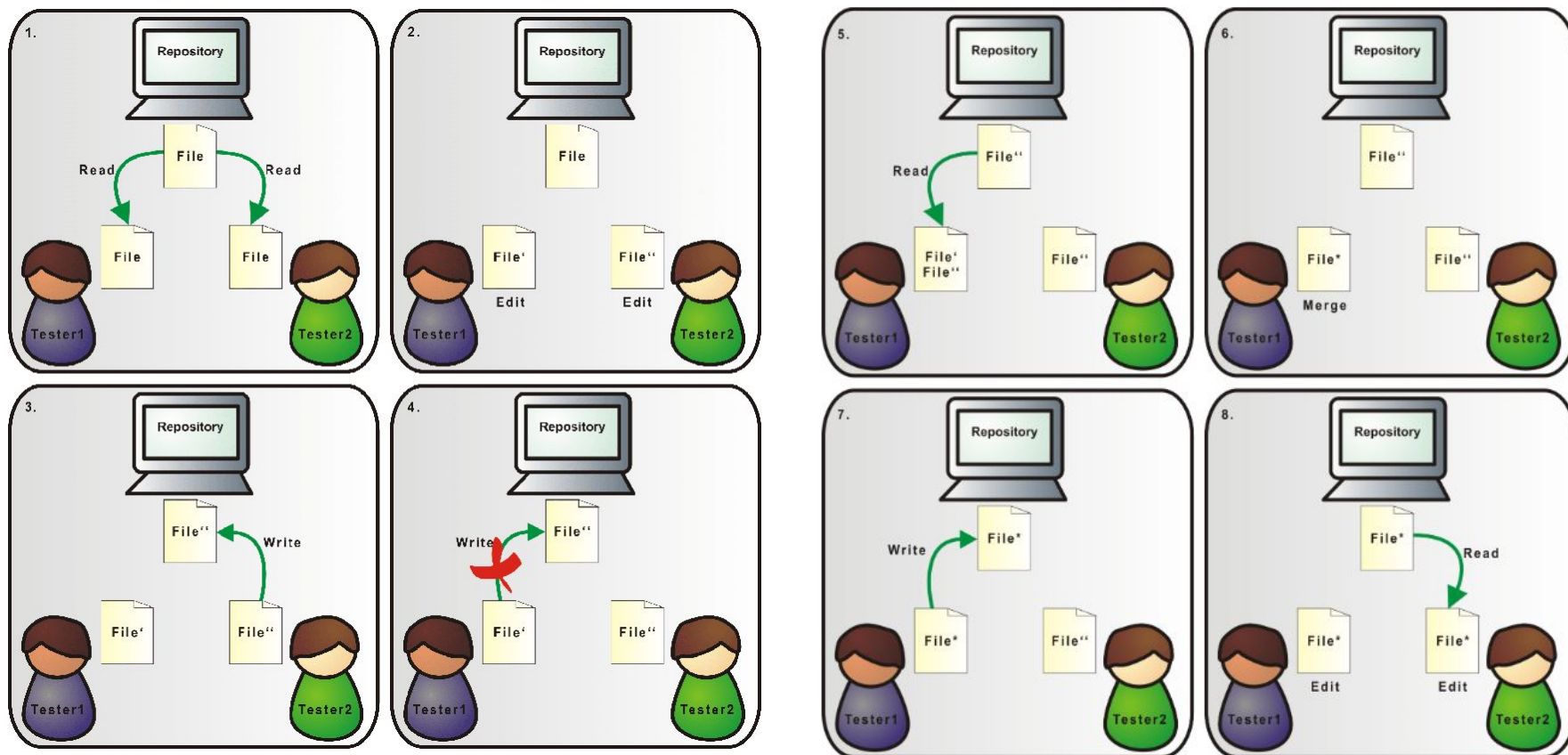
# ВИДЫ VCS: LOCK-MODIFY-UNLOCK

9



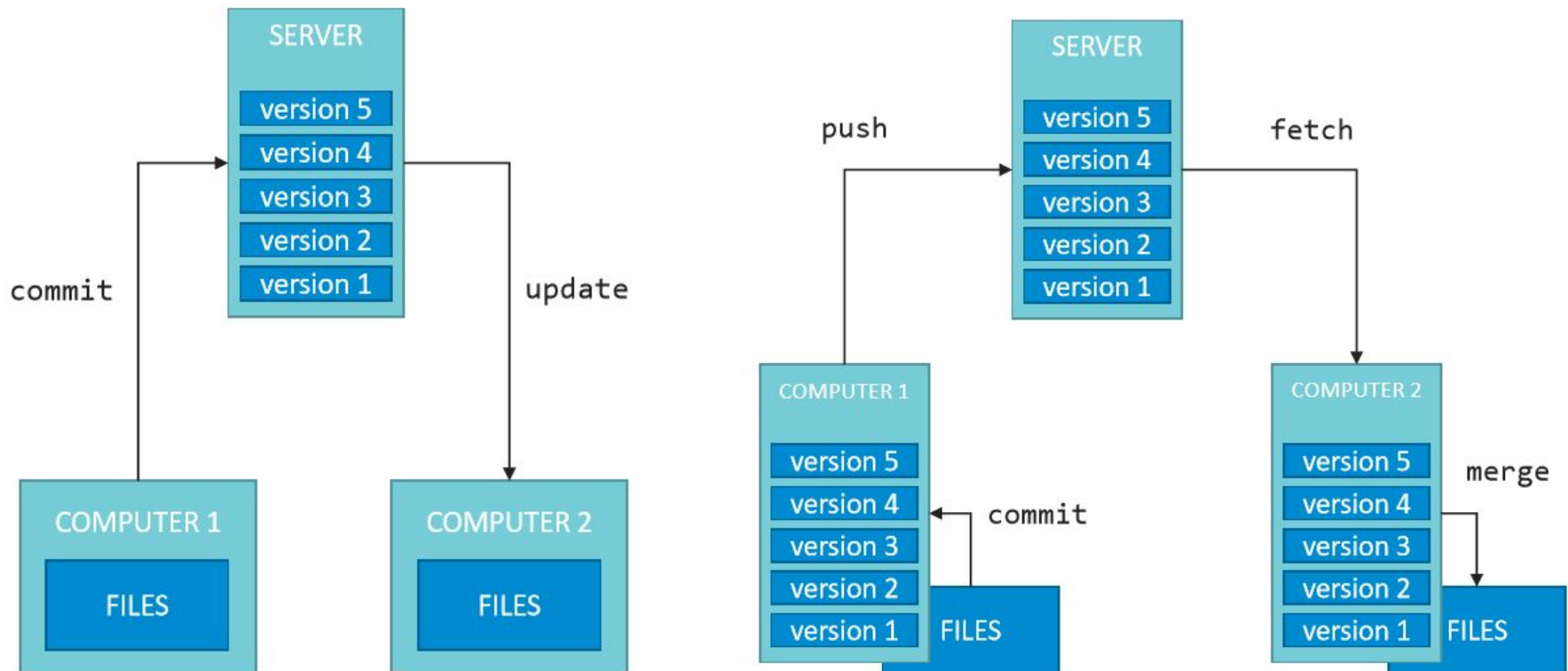
# ВИДЫ VCS: COPY-MODIFY-MERGE

10



# ВИДЫ **VCS**: ЦЕНТРАЛИЗОВАННАЯ ИЛИ РАСПРЕДЕЛЕННАЯ

11



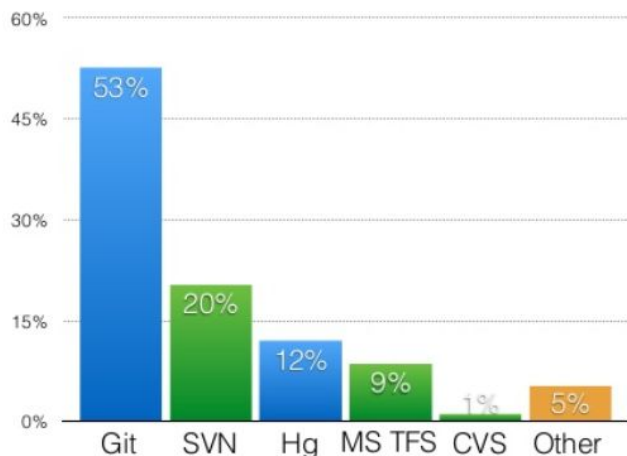
# ПОЧЕМУ GIT

12



В 2005 году Линус Торвалдс разработал GIT для управления версиями ядра Linux, однако, чуть позднее его поддержкой занялся другой человек, японский инженер-программист – Джуньо Хамано. На сегодняшний день, GIT – это одна из самых известных систем контроля версий с открытым исходным кодом, на которую полагаются миллионы проектов по всем миру (включая как коммерческие, так и бесплатные проекты). GIT – это полностью бесплатное программное обеспечение, поддерживающее множество ОС, таких как Mac, Linux, Windows и Solaris. Скачать соответствующие дистрибутивы можно [здесь](#) [или](#) [о сайта GIT.](#)

## Git is the most used VCS



# ПРЕИМУЩЕСТВА GIT

13

- ✓ Распределенная система управления версиями, GIT следует принципу одноранговой сети – **peer to peer** (равный к равному) в отличие от других систем вроде Subversion (SVN), которая основана на модели **client-server** (клиент-сервер).
- ✓ GIT позволяет разработчикам иметь множество совершенно независимых веток кода. Создание, удаление и объединение этих веток происходит без каких-либо проблем и больших затрат времени
- ✓ В GIT все операции атомарны; это означает, что любое действие может быть полностью удачным или провалиться (без каких-либо изменений). Это действительно важно, так как в некоторых системах контроля версий (вроде CVS), где действия не атомарны, некоторые повисшие операции по всему хранилищу, могут оставить его в нестабильном состоянии.
- ✓ В отличие от других VCS, таких как SVN или CVS где метаданные хранятся в скрытых папках (.cvs, .svn, и т.д.), в GIT все данные расположены в каталогах .git.
- ✓ Он использует модель данных, которая помогает обеспечить криптографическую целостность всего, что присутствует в репозитории. Каждый раз когда файлы добавляются или коммитятся, генерируются их контрольные суммы; аналогичный процесс происходит при их извлечении.
- ✓ Еще одна превосходная функция, присутствующая в GIT – это его **индекс**. В пределах индекса, разработчики могут форматировать коммиты и просматривать их до фактического применения.

# УСТАНОВКА И НАСТРОЙКА GIT

14

1. Скачать установочный файл: <http://git-scm.com/downloads>
2. Выполнить все шаги, используя дефолтные опции

Command Prompt

```
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ihar.pryshchepau>git version
git version 2.16.1.windows.1
```

3. Открыть консоль git bash и назначить имя и емейл вашего пользователя

- ❑ \$ MINGW64 ~ git config --global user.name "Ivan Petrov"
- ❑ \$ MINGW64 ~ git config --global user.email [ivan.petrov@gmail.com](mailto:ivan.petrov@gmail.com)

4. Сгенерировать ssh ключ для работы с репозиторием по протоколу ssh

- ❑ \$ MINGW64 ~ ssh-keygen -t rsa -C "[ivan.petrov@gmail.com](mailto:ivan.petrov@gmail.com)"

c:\Users\ihar.pryshchepau\.ssh\*.*					* ▼
↑ Name	Ext	Size	Date	Attr	
⌚ [..]		<DIR>	08/14/2019 13:08	----	
📄 id_rsa		3,326	08/14/2019 13:08	-a--	
📄 id_rsa	pub	746	08/14/2019 13:08	-a--	
📄 known_hosts		187	08/14/2019 12:53	-a--	



# GITLAB REPOSITORY: CREATE & CLONE

15

1. Создать аккаунт в: <https://gitlab.com>
2. Добавить в профиль ssh public key

The screenshot shows the GitLab web interface. The browser address bar displays `gitlab.com/profile/keys`. The top navigation bar includes links for Projects, Groups, Activity, Milestones, Snippets, and Analytics. The left sidebar shows the 'User Settings' menu with options like Profile Settings, Account, Billing, Applications, Chat, Access Tokens, Emails, Password, Notifications, SSH Keys (highlighted), GPG Keys, and Preferences. The main content area is titled 'SSH Keys' and explains that SSH keys allow for a secure connection between a computer and GitLab. It provides instructions on how to add a new SSH key, either by generating one or using an existing one. A large text area is provided for pasting the public SSH key, with a hint that it typically starts with 'ssh-ed25519 ...' or 'ssh-rsa ...'. Below the text area, there is a 'Title' field with the placeholder text 'e.g. My MacBook key' and a note to name the key individually. An 'Add key' button is located at the bottom of the form.

gitlab.com/profile/keys

GitLab Projects Groups Activity Milestones Snippets Analytics

User Settings

Profile Settings

Account

Billing

Applications

Chat

Access Tokens

Emails

Password

Notifications

SSH Keys

GPG Keys

Preferences

User Settings > SSH Keys

### SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

#### Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

#### Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id\_ed25519.pub' or '~/.ssh/id\_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."

#### Title

e.g. My MacBook key

Name your individual key via a title

Add key



## 3. Создать новый проект(репозиторий)

Blank project	Create from template	Import project	CI/CD for external repo
---------------	----------------------	----------------	-------------------------

Project name

My awesome project

Project URL

https://gitlab.com/ iproff2011

Project slug

my-awesome-project

Want to house several dependent projects under the same namespace? [Create a group.](#)

Project description (optional)

Description format

Visibility Level ?

☒ Private  
Project access must be granted explicitly to each user.

☐ Public  
The project can be accessed without any authentication.

☐ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project

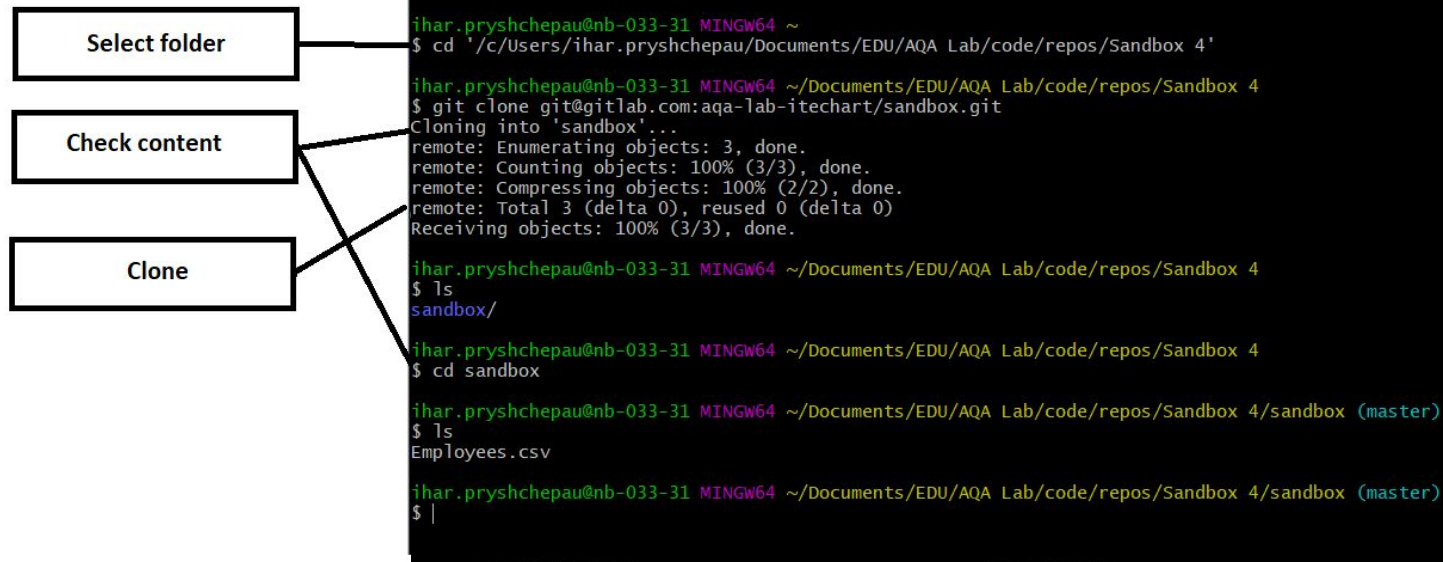
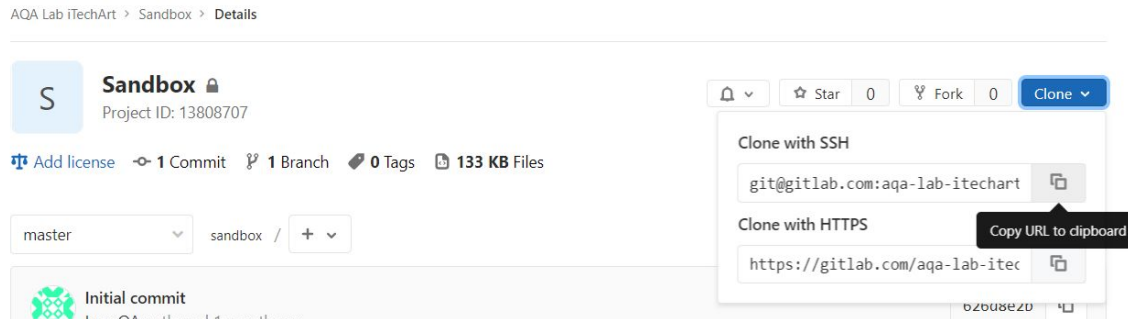
Cancel



# GITLAB REPOSITORY: CREATE & CLONE

17

## 4. Скопировать в буфер обмена адрес репозитория для ssh



# GIT: COMMIT AND PUSH

18

- `$ git status` – показывает текущее состояние локального репозитория
- `$ git add <file name>` – добавляет изменения в git index
- `$ git commit -m <message>` – создает commit с комментарием в локальном репозитории
- `$ git push` – отправляет все существующие коммиты в локальном репозитории в удаленный репозиторий в соответствующую ветку

```
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        tobeornottobe.txt

nothing added to commit but untracked files present (use "git add" to track)

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git add tobeornottobe.txt

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   tobeornottobe.txt

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git commit -m "Add file tobeornottobe.txt with 2 lines"
[master 4fcefca] Add file tobeornottobe.txt with 2 lines
1 file changed, 2 insertions(+)
create mode 100644 tobeornottobe.txt

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 377 bytes | 377.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To gitlab.com:iproff2011/poem.git
   0f7a9a5..4fcefca  master -> master

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ |
```

# GIT: FETCH FROM REMOTE AND MERGE

19

- \$ git fetch – собирает все коммиты из целевой ветки, которых нет в текущей ветке, и сохраняет их в локальном репозитории, не сливая их в текущую ветку.
- \$ git merge – добавляет изменения в текущую ветку

```
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From gitlab.com:iproff2011/poem
 17129fe..851a12b  master    -> origin/master

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git merge
Updating 17129fe..851a12b
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

# GIT: PULL FROM REMOTE

20

- \$ git pull – добавляет все коммиты в текущую ветку без возможности их просмотра

```
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

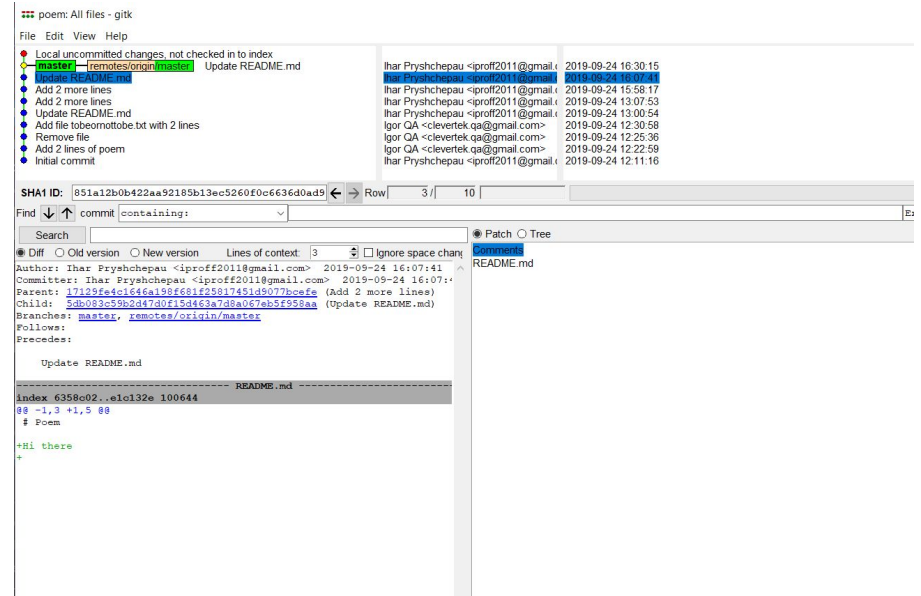
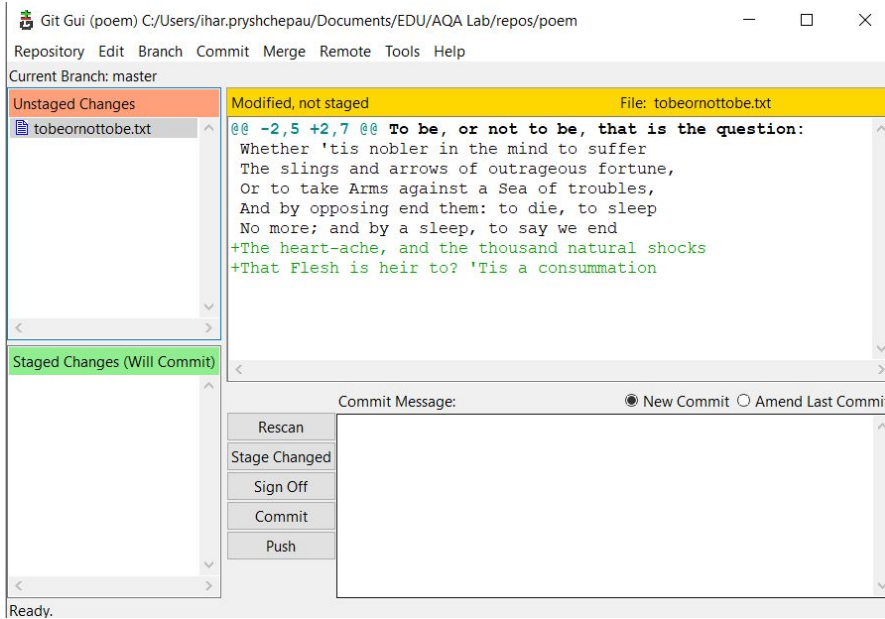
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From gitlab.com:iproff2011/poem
   851a12b..5db083c master    -> origin/master
Updating 851a12b..5db083c
Fast-forward
 README.md | 6 +-----
 1 file changed, 1 insertion(+), 5 deletions(-)

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/repos/poem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

# GIT GUI & GITK

21





## Working directory

```
git checkout -- file.txt  
git checkout .  
git clean -xdf
```

## Staging area (Index)

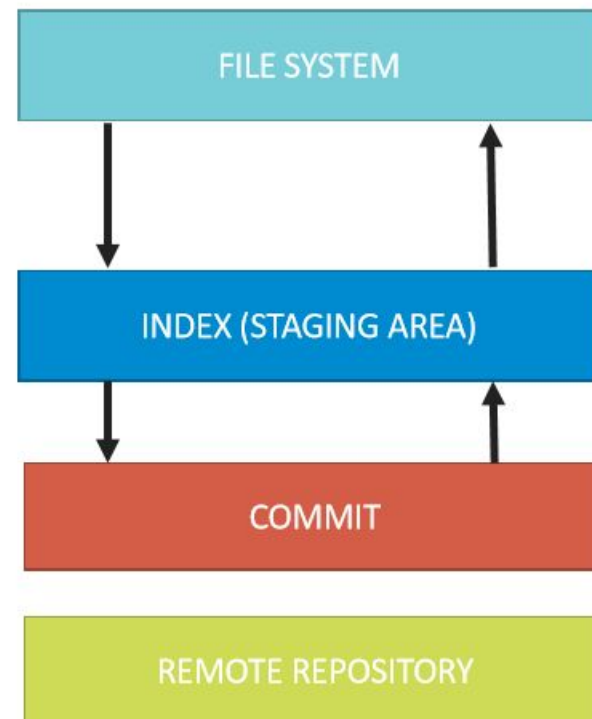
```
git reset -- file.txt
```

## Local branch

```
git reset HEAD^^ (HEAD~2)  
git commit --amend -m "commit message"
```

## Remote repository

```
git revert <sha1>
```



## .gitignore

---

```
# no .log files
*.log

# but do track error.log, even though you're
# ignoring .log files above
!error.log

# only ignore the TODO file in the current
# directory, not subdir/TODO
/TODO

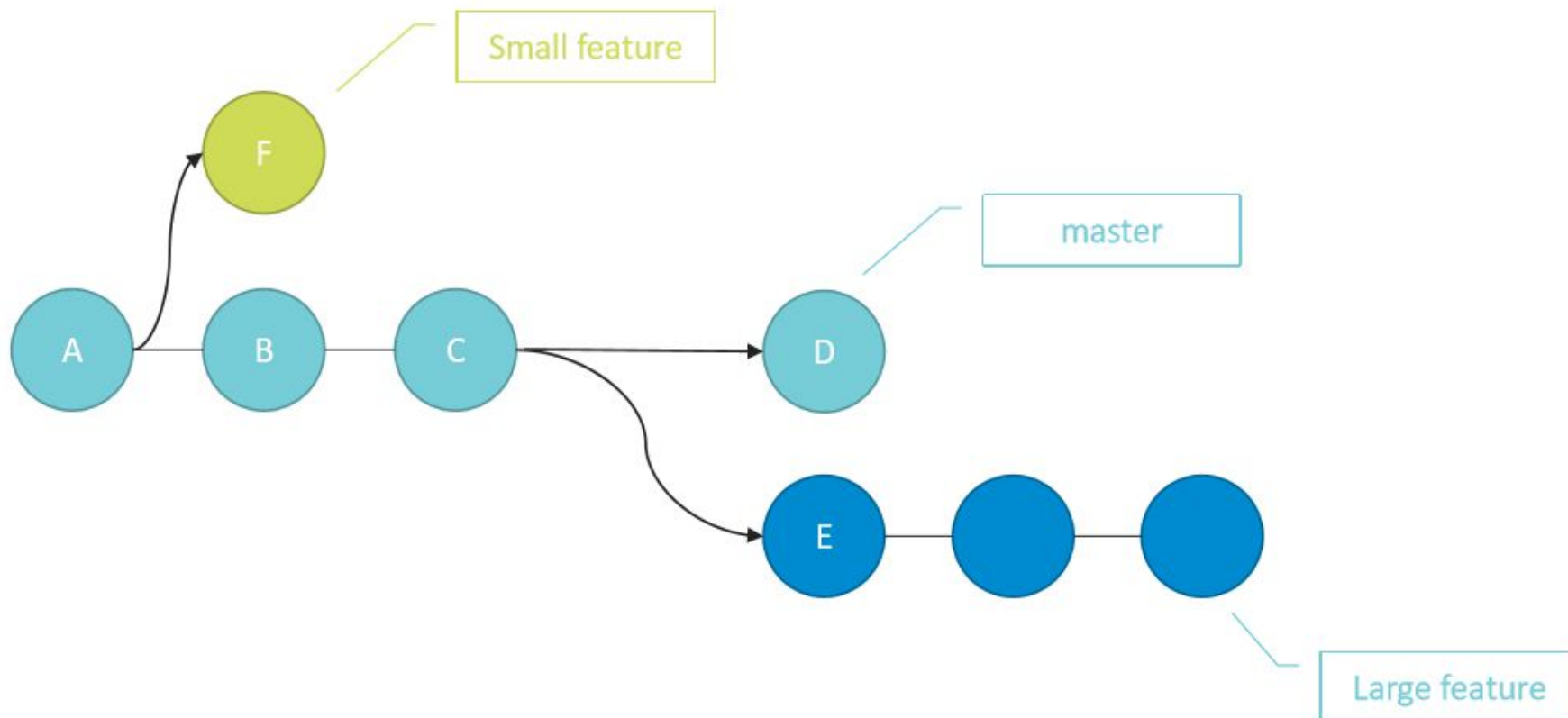
# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory
doc/**/*.pdf
```

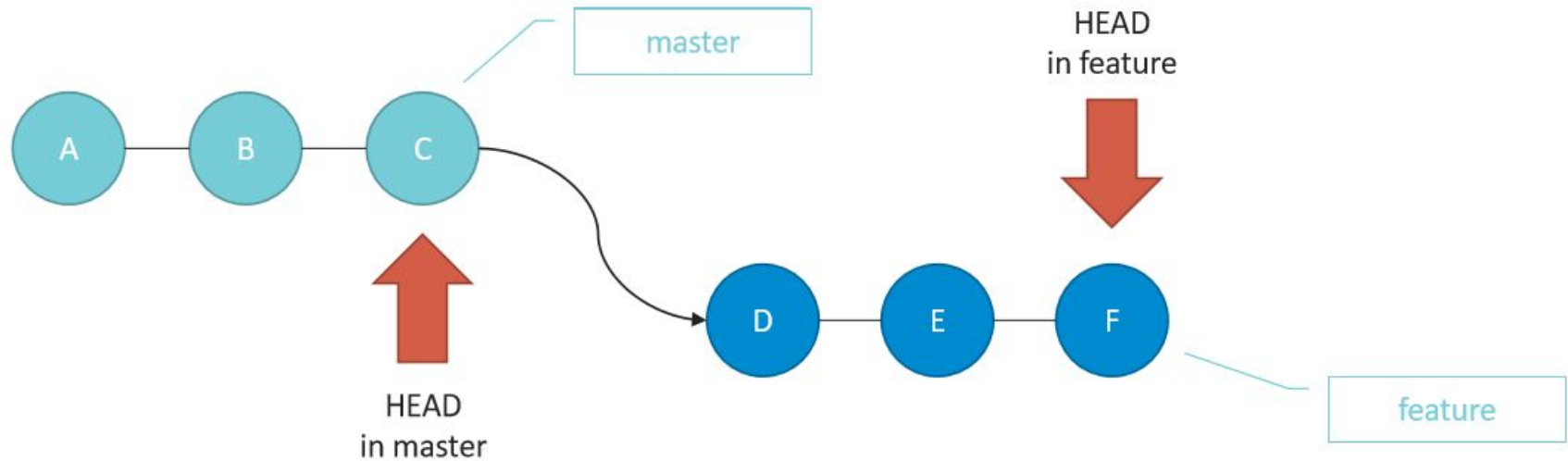
# GIT: BRANCH CONCEPT

24





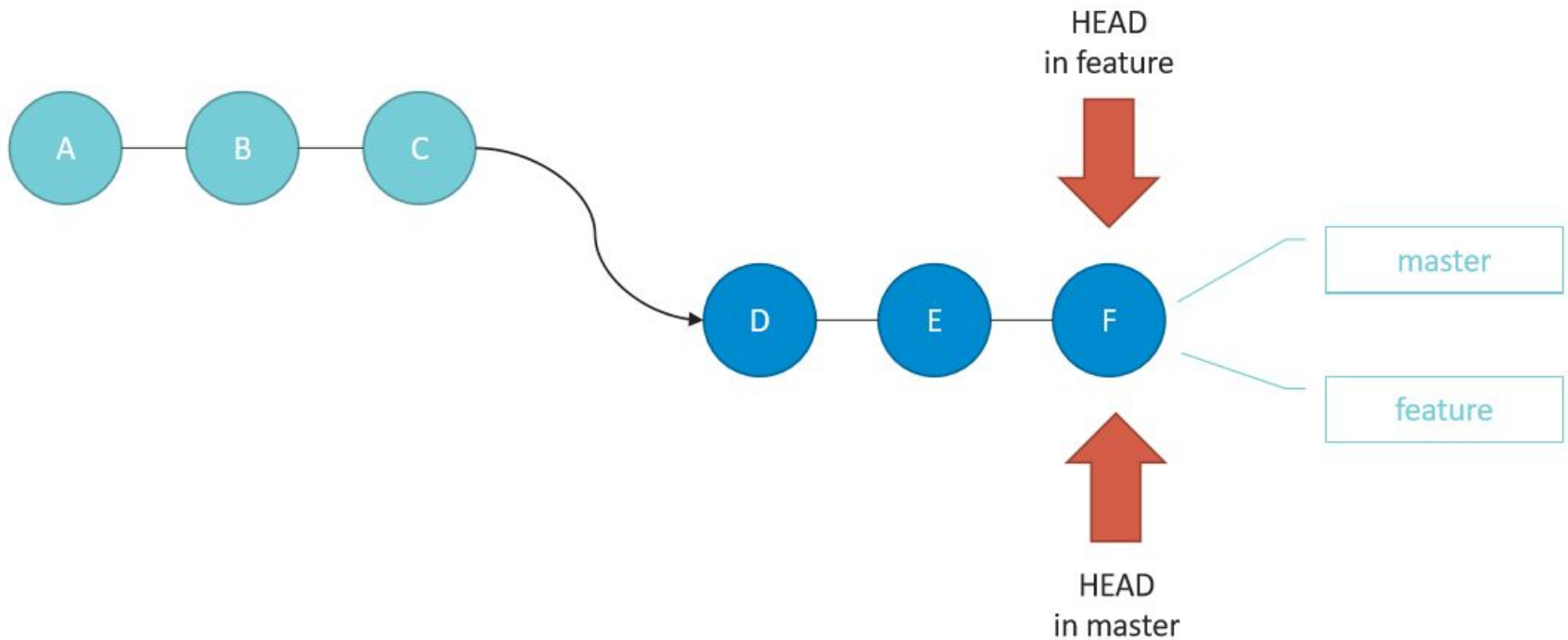
## Fast-forward merge



# GIT: BRANCH MERGE

26

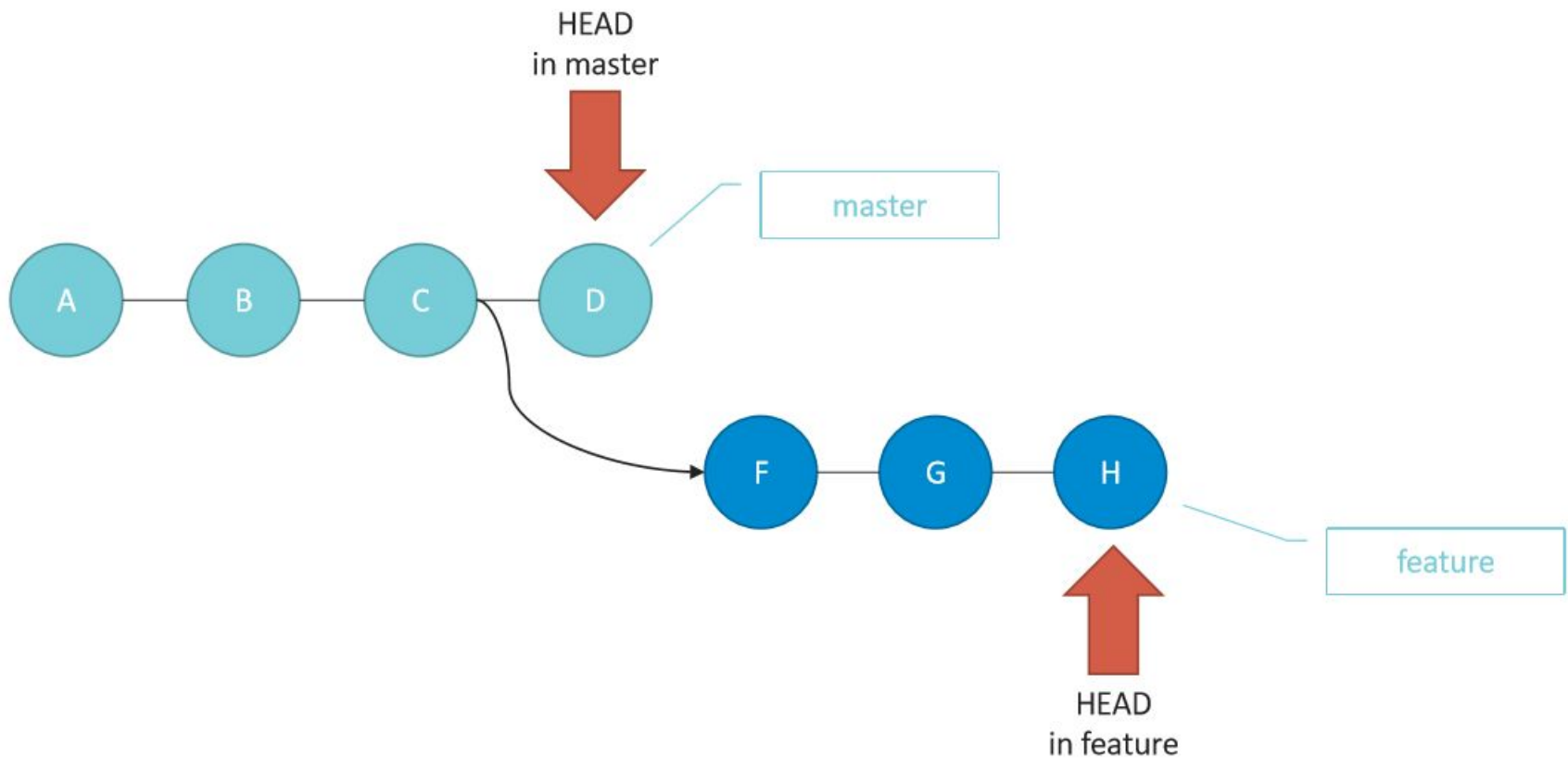
Fast-forward merge



# GIT: BRANCH MERGE

27

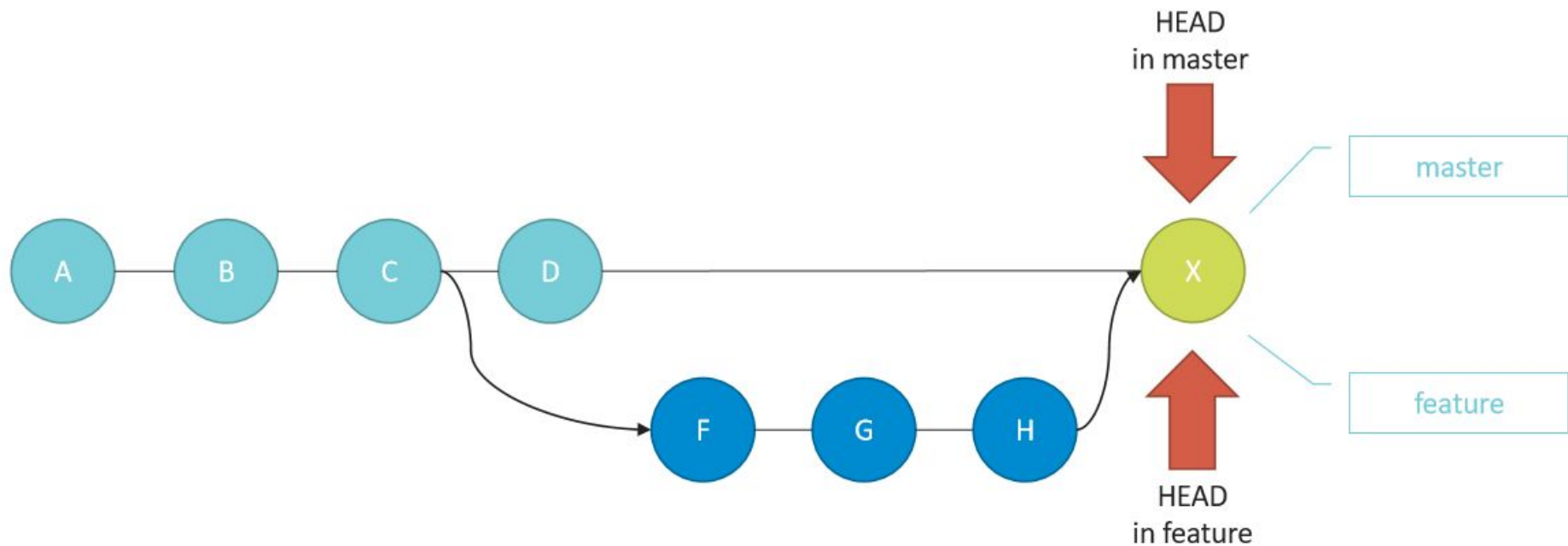
## Non fast-forward merge



# GIT: BRANCH MERGE

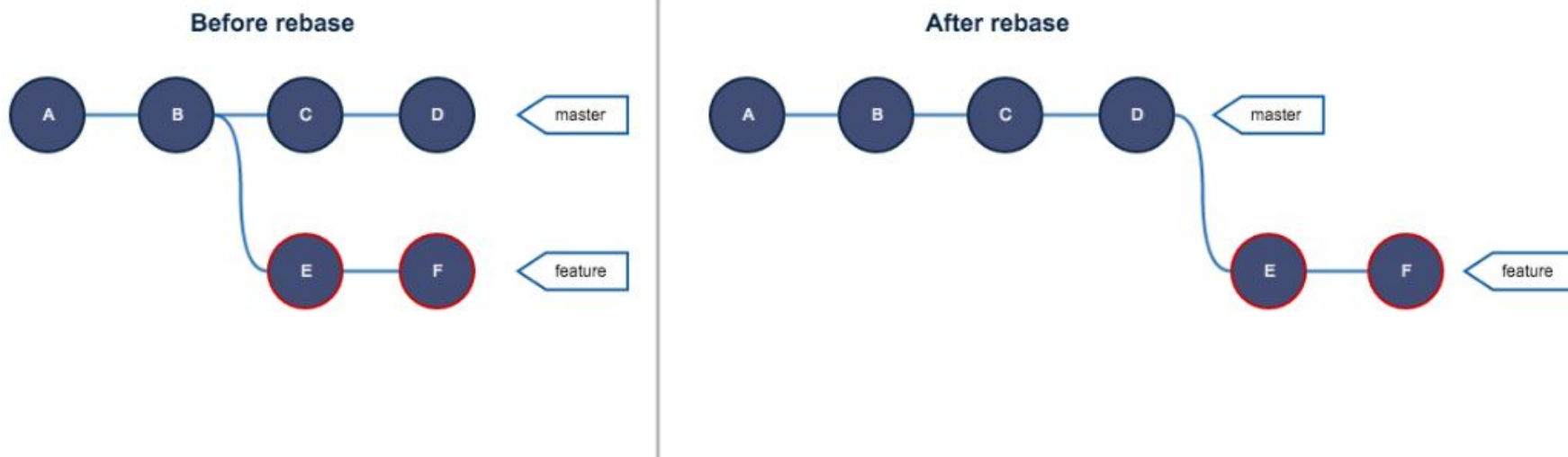
28

Non fast-forward merge



# GIT: BRANCH REBASE

29



**ЗОЛОТОЕ ПРАВИЛО:  
!!!НИКОГДА НЕ ДЕЛАЙТЕ REBASE ОБЩИХ ВЕТОК**

## SOLVE CONFLICT

Abort merge

```
git merge --abort
```

Resolve by selecting version

```
git checkout --ours <branch name> <file name>
```

Resolve manually

```
git diff
```

Undo merge

```
git revert 09fe472
```

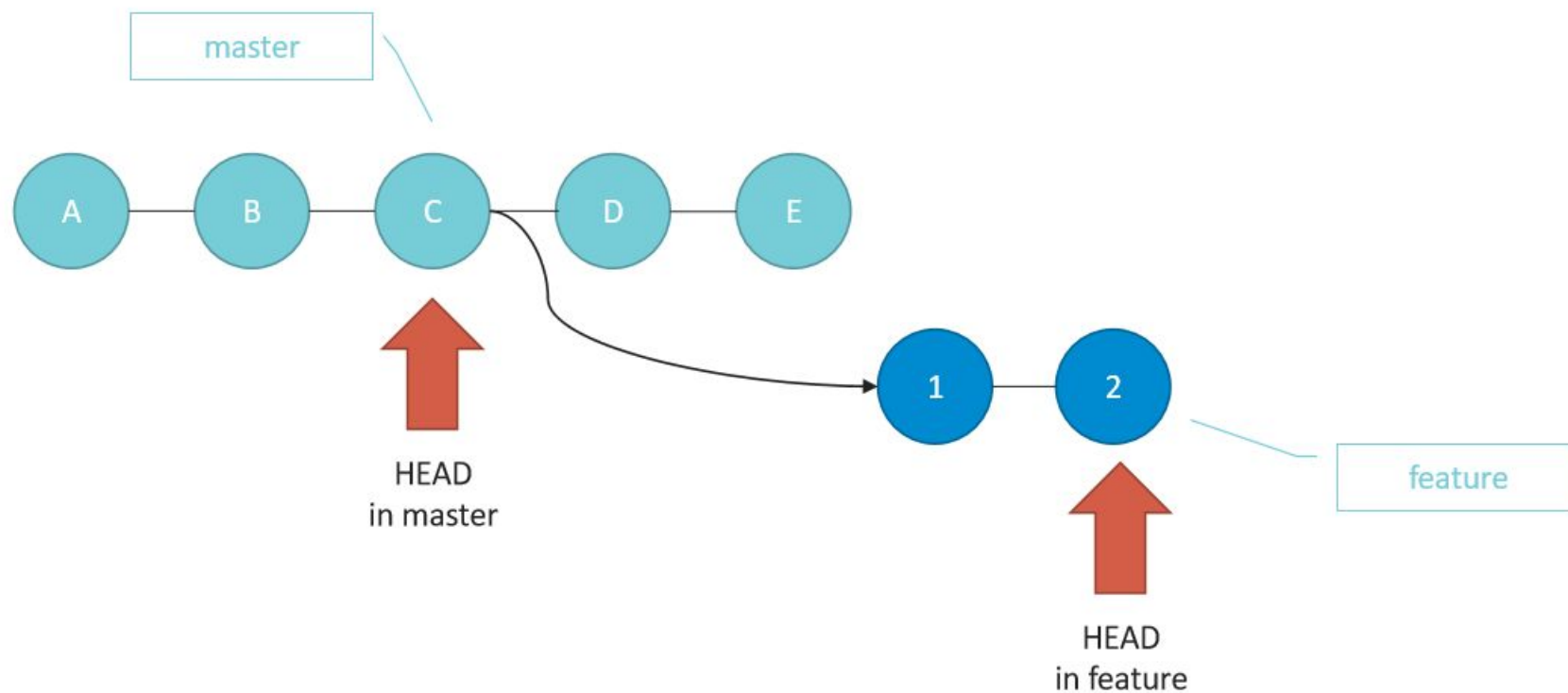
User merge tool

## AVOID CONFLICT

- Short commits
- No edits to whitespaces
- Merge often

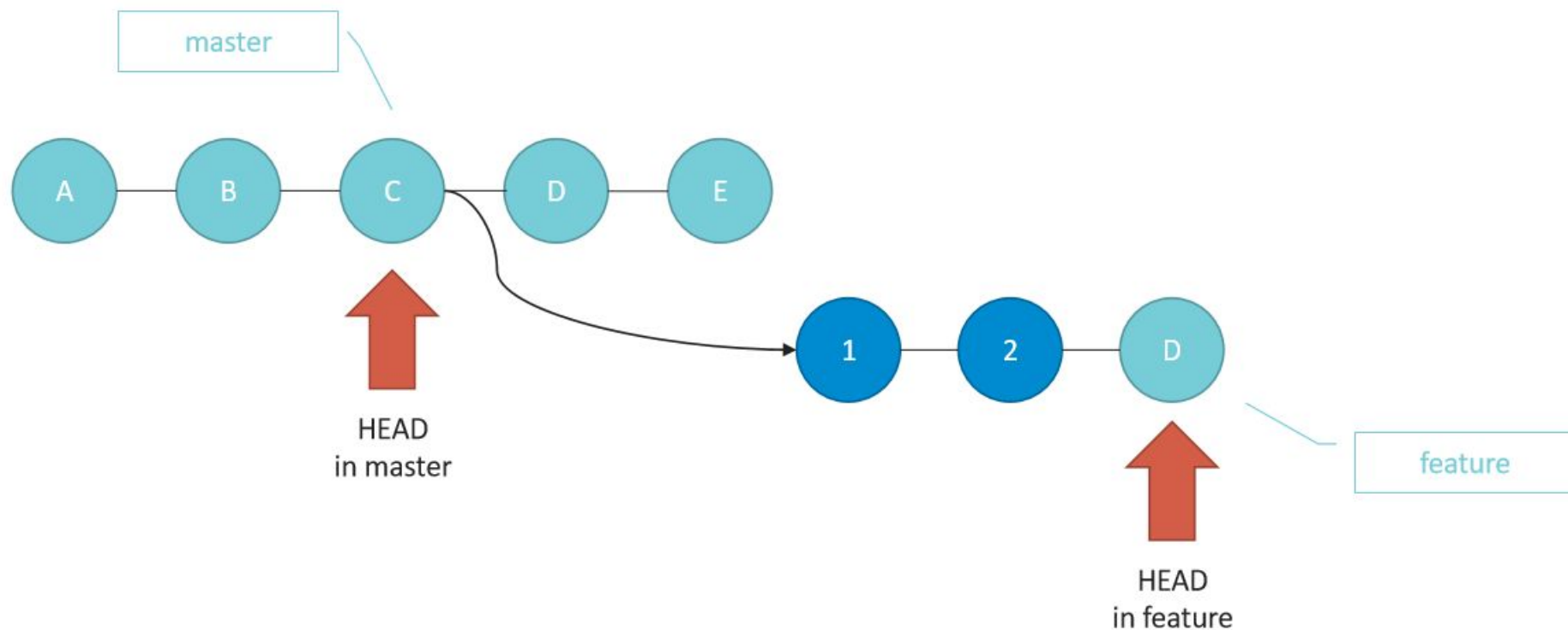
# GIT: CHERRY PICK

31

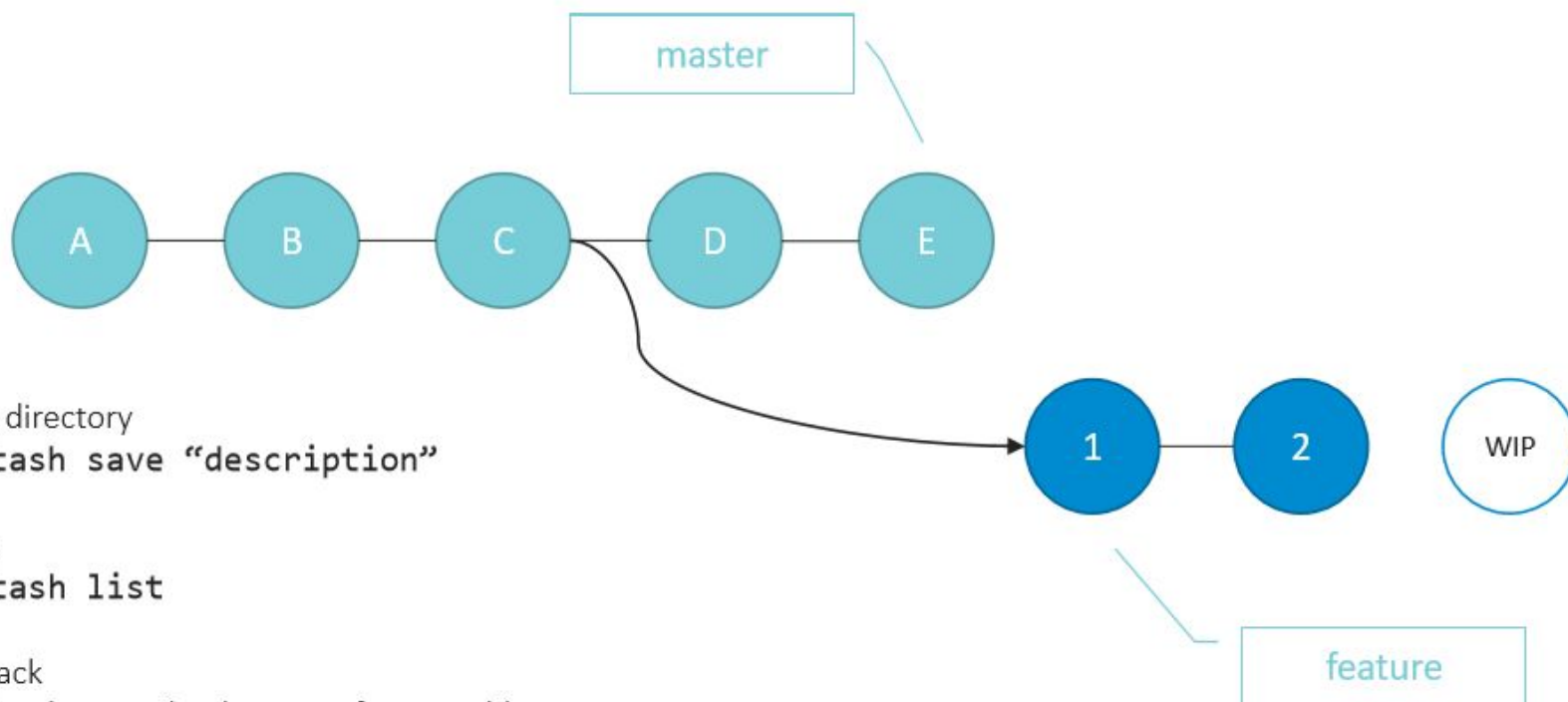


# GIT: CHERRY PICK

32







Save working directory

```
git stash save "description"
```

View stashes

```
git stash list
```

Bring them back

```
git stash pop (and remove from stash)
```

```
git stash apply (leave in stash)
```

Remove

```
git stash drop (clear)
```

# GIT: ADD LOCAL PROJECT TO REMOTE

34

- \$ git init – инициализирует текущую папку как локальный репозиторий git

```
MINGW64:/c:/Users/ihar.pryshchepau/Documents/EDU/AQA Lab/code/repos/...
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/code/repos/GitLab2
$ git init
Initialized empty Git repository in C:/Users/ihar.pryshchepau/Documents/EDU/AQA Lab/code/repos/GitLab2/.git/

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/code/repos/GitLab2 (master)
$ git add .

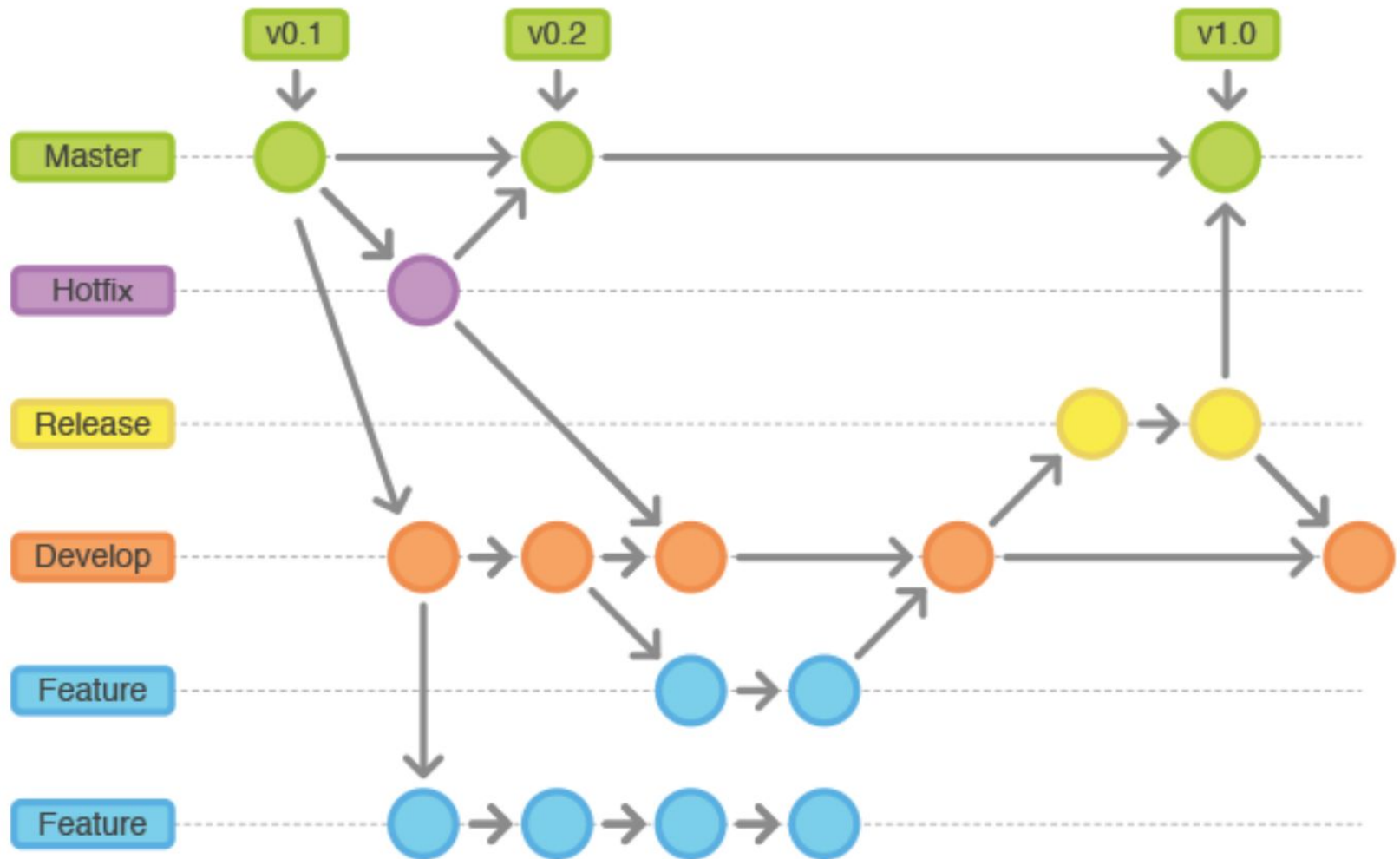
ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/code/repos/GitLab2 (master)
$ git commit -m "Add files to the new repository"
[master (root-commit) 6a0489d] Add files to the new repository
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 my_project.txt

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/code/repos/GitLab2 (master)
$ git push --set-upstream git@gitlab.com:iproff2011/example.git master
Counting objects: 3, done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To gitlab.com:iproff2011/example.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'git@gitlab.com:iproff2011/example.git'.

ihar.pryshchepau@nb-033-31 MINGW64 ~/Documents/EDU/AQA Lab/code/repos/GitLab2 (master)
$ |
```

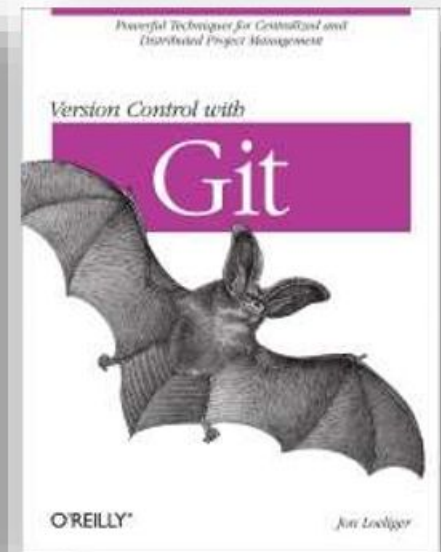
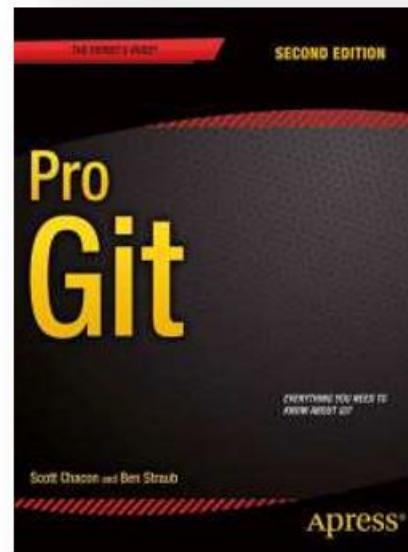
# GIT FLOW

35



- `git config --global user.name "<user name>"`
- `git config --global user.email "vitali_shulha@epam.com"`
- `git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe'"`
- `git blame`
- `git bisect`
- `git log --pretty=oneline`
- `git log --pretty=format:"%h %s" --graph`
- `git config --global alias.last 'log -1 HEAD'`
- `git last`
- `git log master..experiment`
- `git filter-branch --tree-filter 'rm -f passwords.txt' HEAD`
- `git rerere`
- `git submodule`

- **Pro Git** by Scott Chacon and Ben Straub
- **Version Control with Git** by Jon Loeliger, Matthew McCullough



# In case of fire



1. `git commit`



2. `git push`



3. leave building