

Системы контроля версий

Выполнил Горвиц Евгений, ВМИ-301

**Что такое система
контроля версий?**

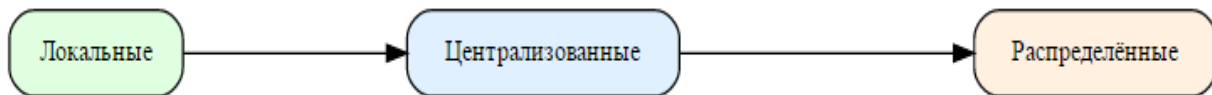
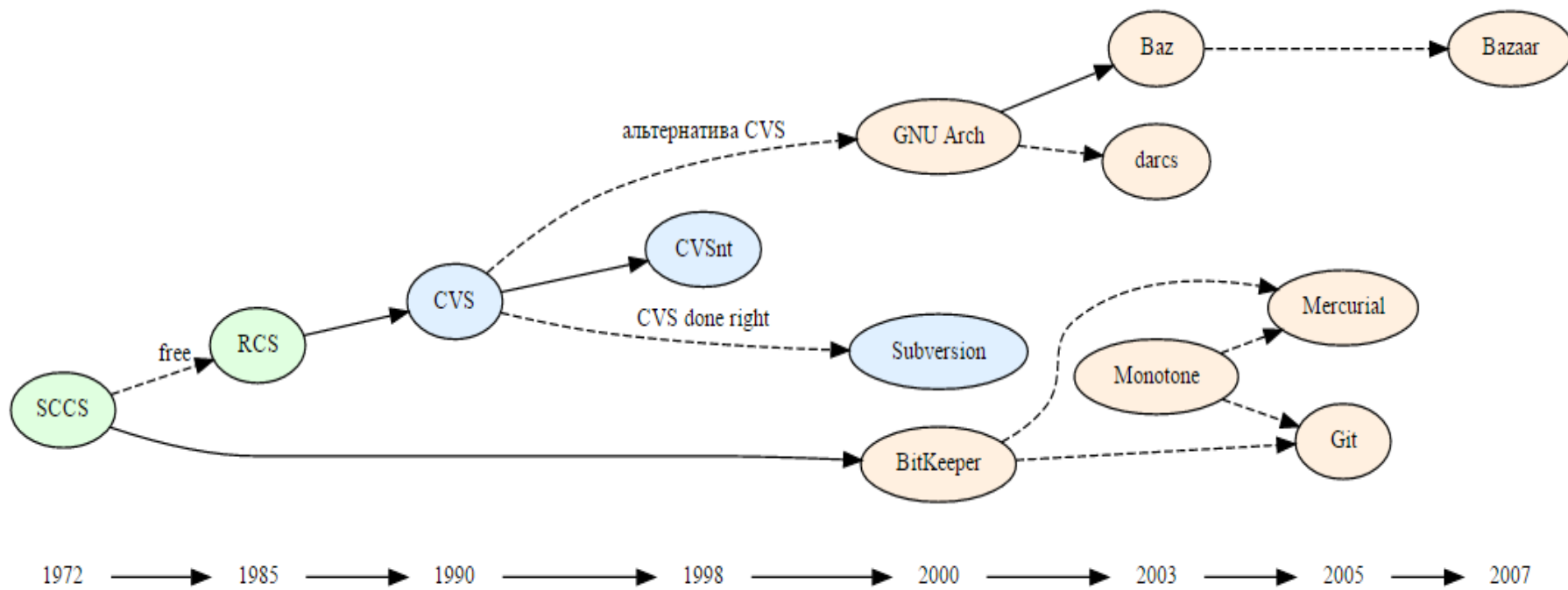
- ▶ **Система контроля версий** (*Version Control System, VCS*) — программное обеспечение для облегчения работы с изменяющейся информацией.
- ▶ VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.



Для чего нужны VCS?

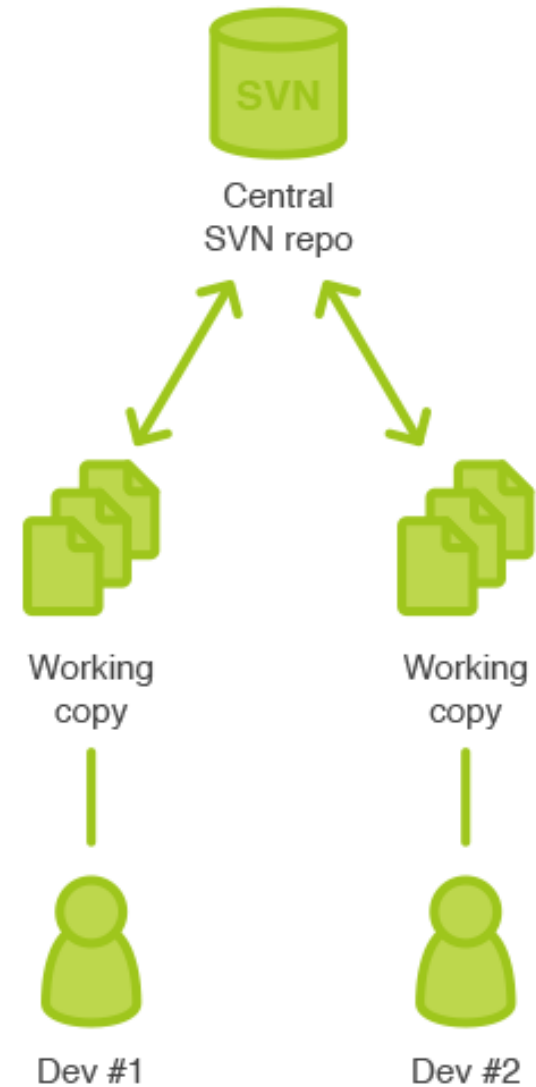
- ▶ Хранение полной истории изменений
- ▶ Описание причин всех производимых изменений
- ▶ Откат изменений, если что-то пошло не так
- ▶ Поиск причины и ответственного за появления ошибок в программе
- ▶ Совместная работа группы над одним проектом
- ▶ Возможность изменять код, не мешая работе других пользователей

История развития

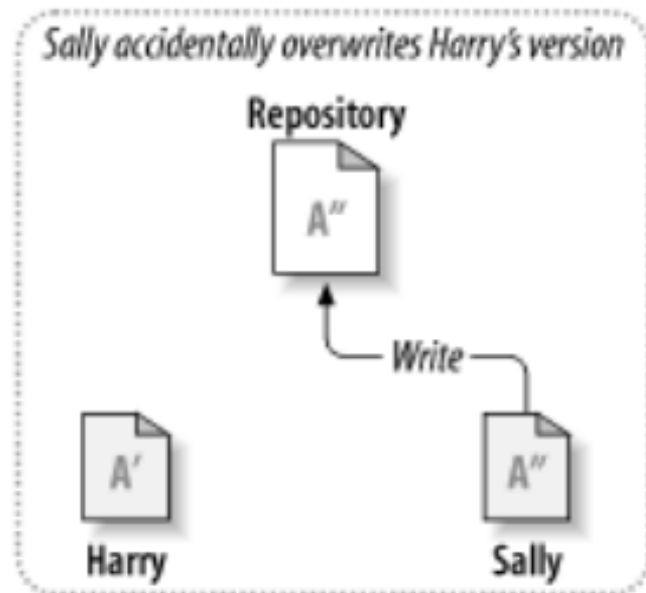
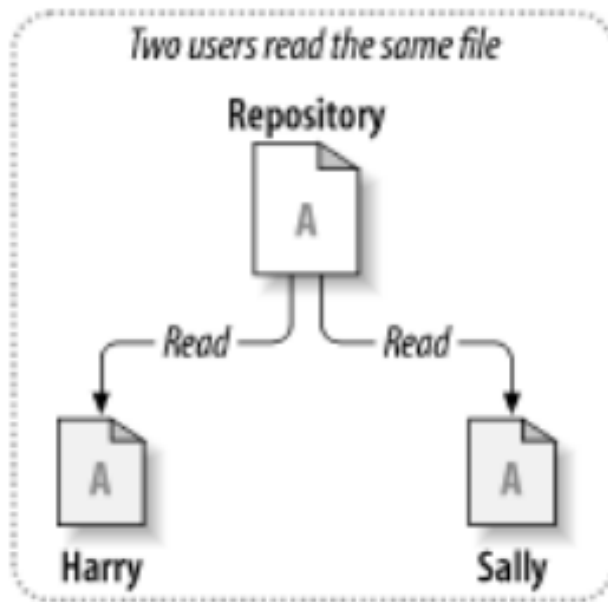


Терминология VCS

- ▶ Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.
- ▶ Рабочая копия - копия проекта, связанная с репозиторием

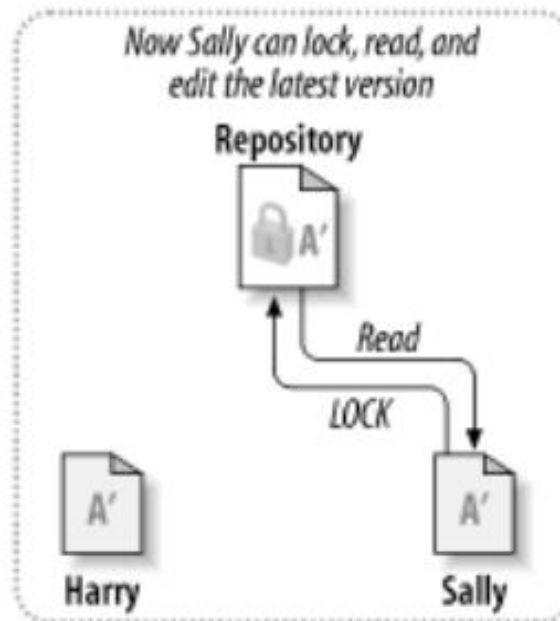
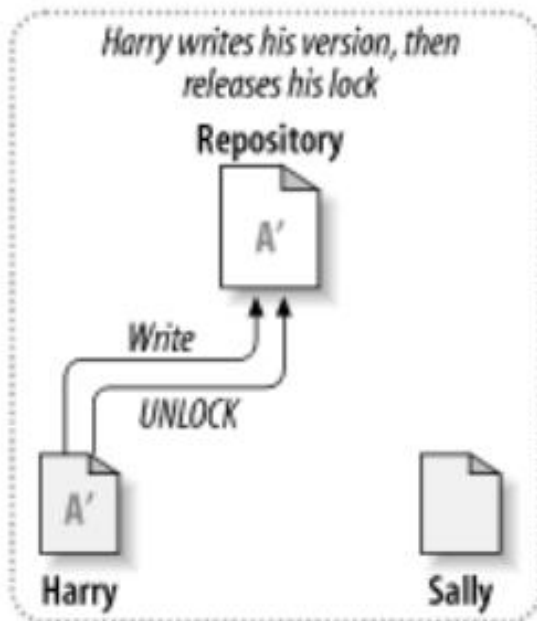
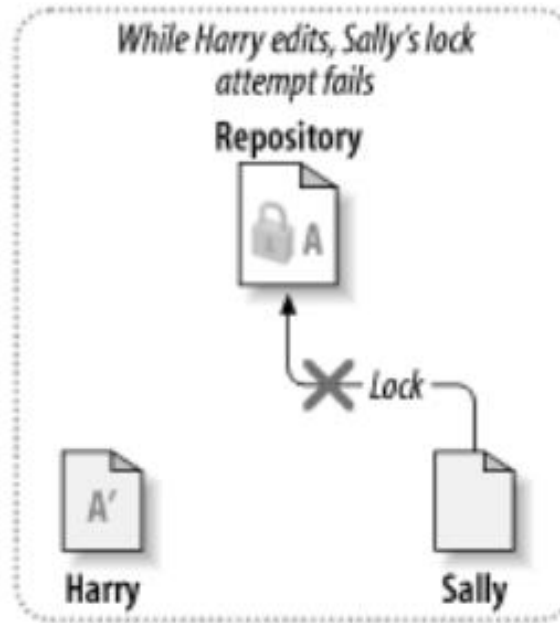
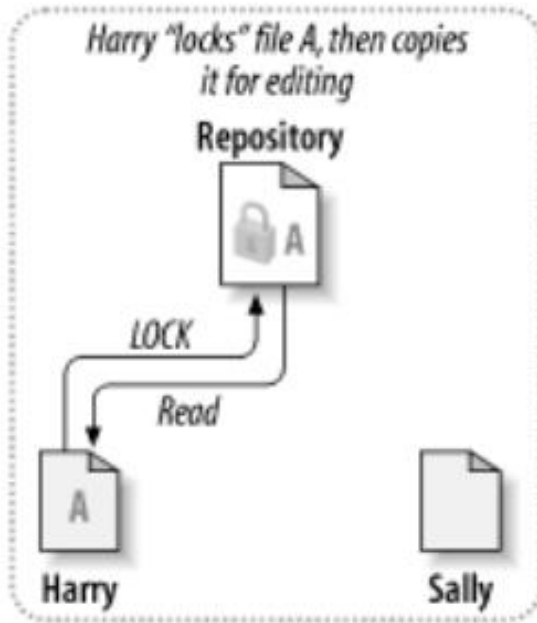


Проблема совместного изменения файлов



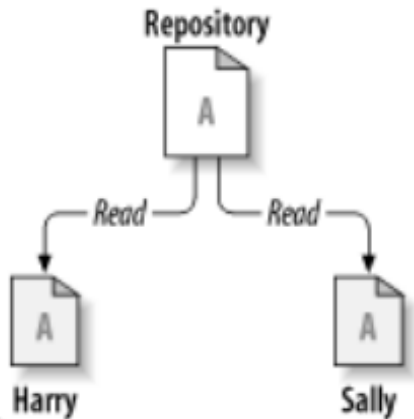
Способы сохранения актуальности данных

Lock-Modify-Unlock

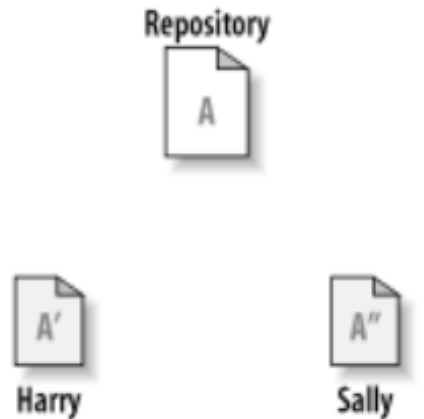


Copy-Modify-Merge

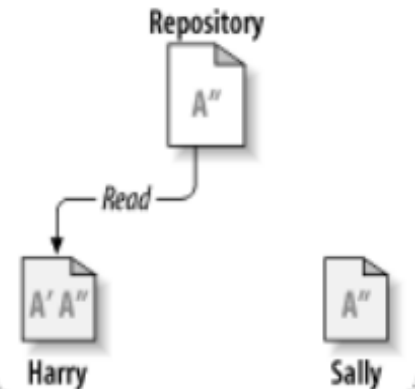
Two users copy the same file



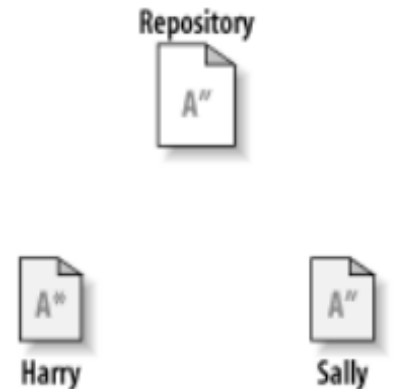
They both begin to edit their copies



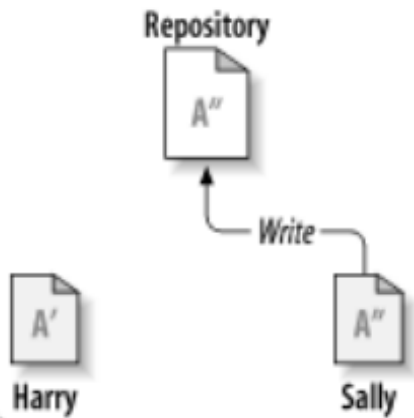
Harry compares the latest version to his own



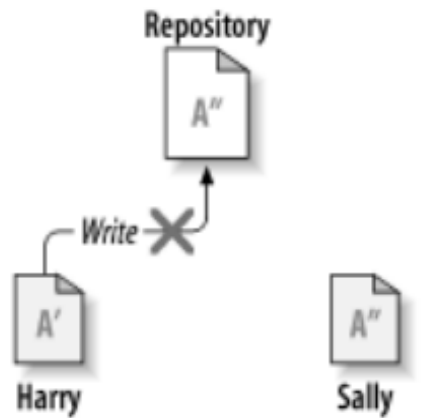
A new merged version is created



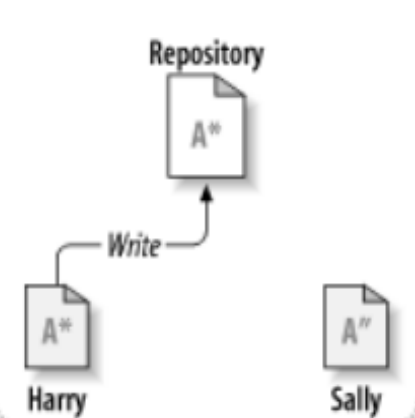
Sally publishes her version first



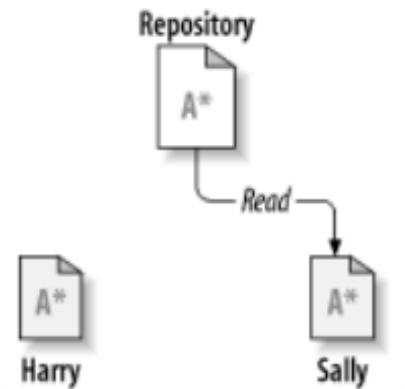
Harry gets an "out-of-date" error



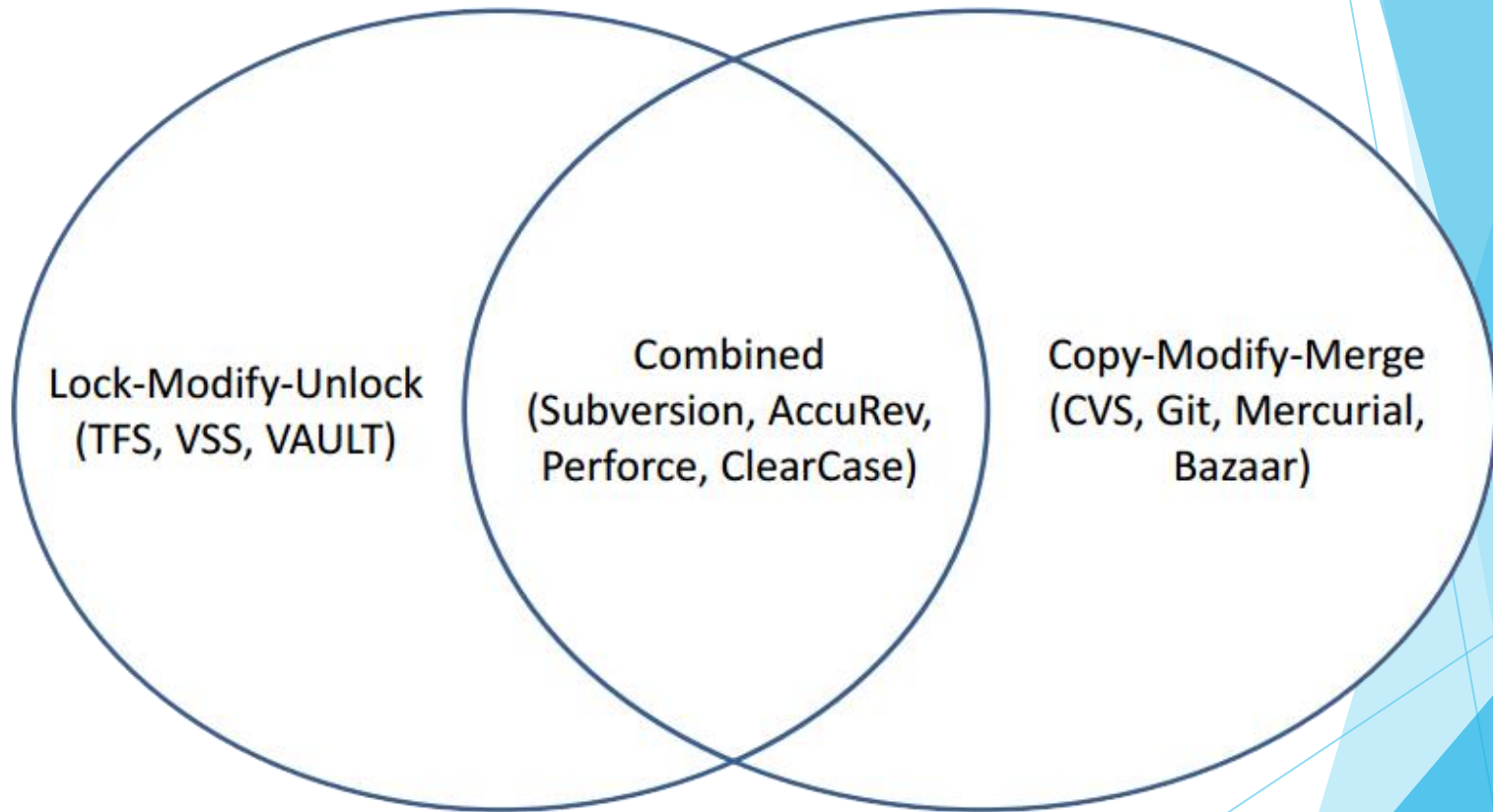
The merged version is published



Now both users have each others' changes



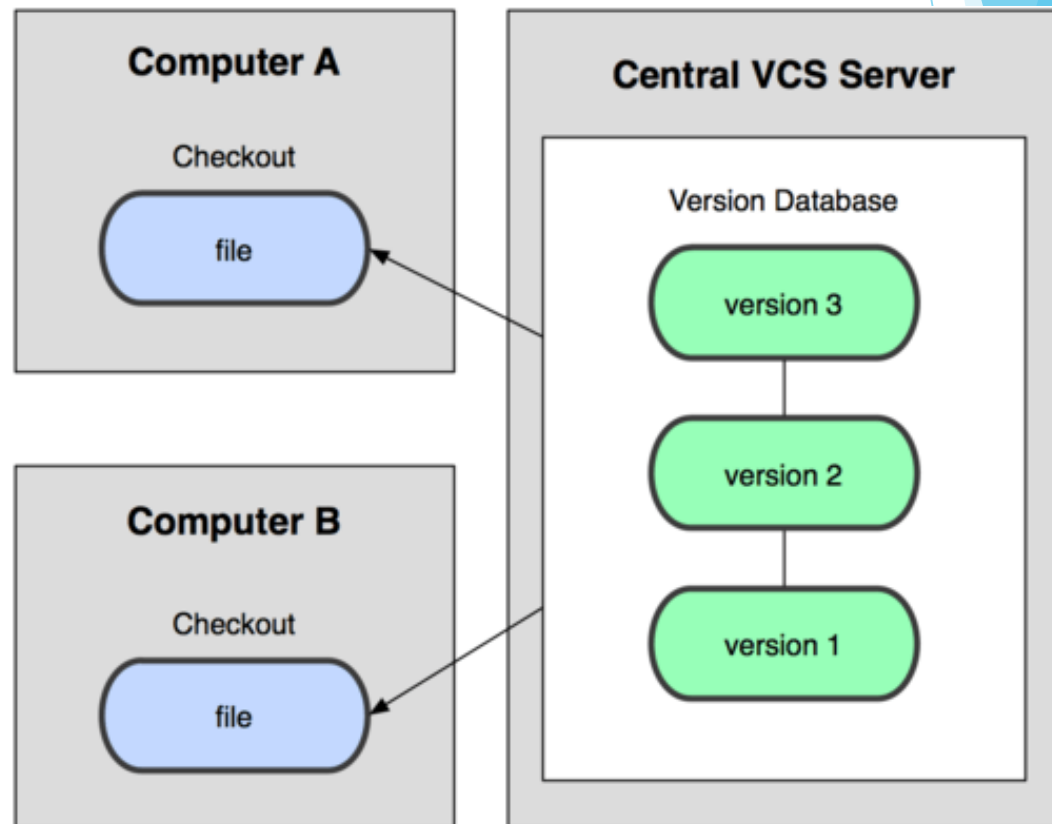
Классификация VCS по способу сохранения актуальности данных



Классификация VCS по способу хранения данных

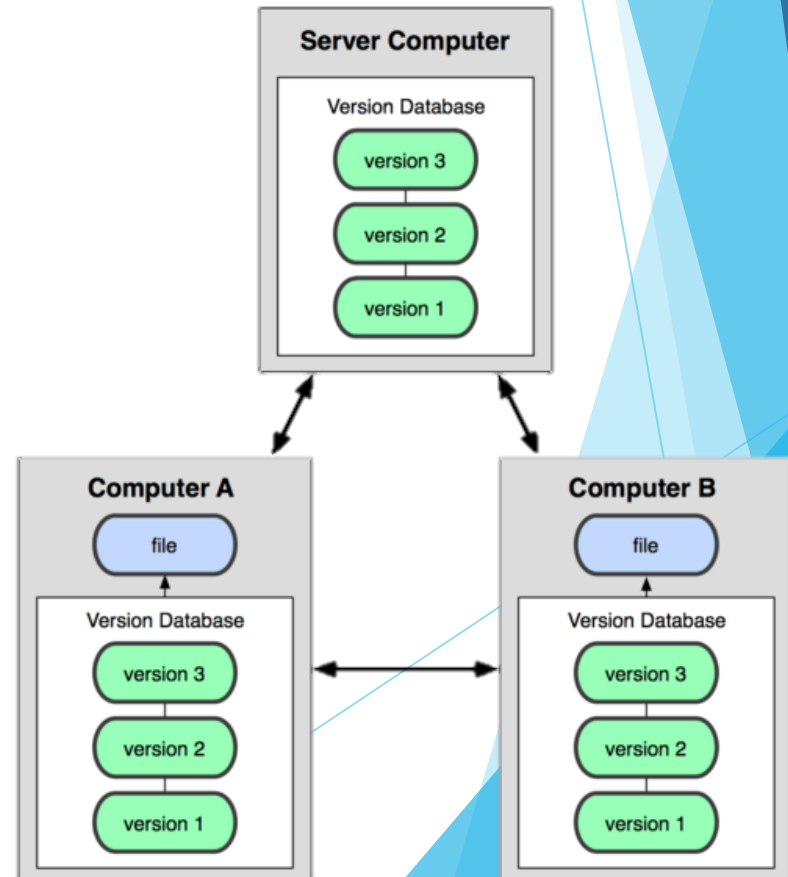
Централизованные VCS

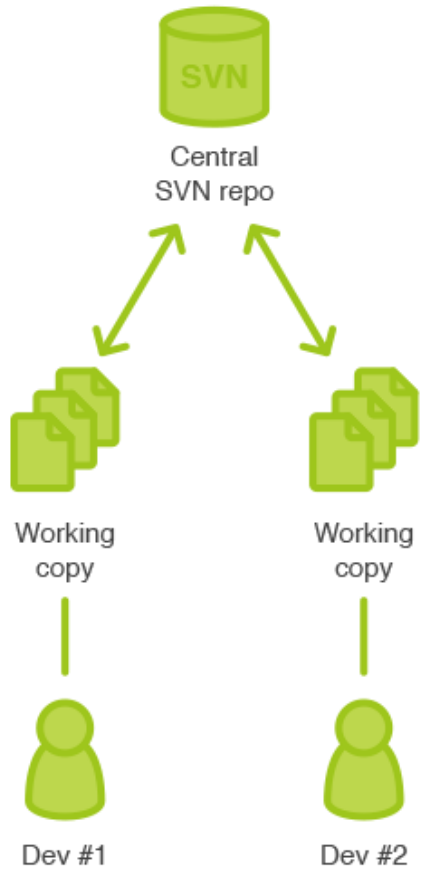
- ▶ Одно основное хранилище всего проекта
- ▶ Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно
 - ▶ Subversion
 - ▶ CVS
 - ▶ TFS, VAULT
 - ▶ AccuRev



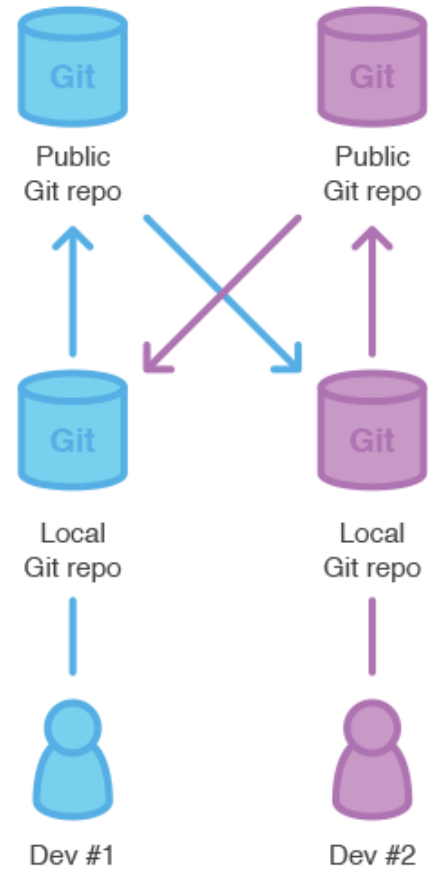
Децентрализованные VCS

- ▶ У каждого пользователя свой вариант (возможно не один) репозитория
- ▶ Присутствует возможность добавлять и забирать изменения из любого репозитория
 - ▶ Git
 - ▶ Mercurial
 - ▶ Bazaar





**Centralized
SVN development**



**Distributed
Git development**

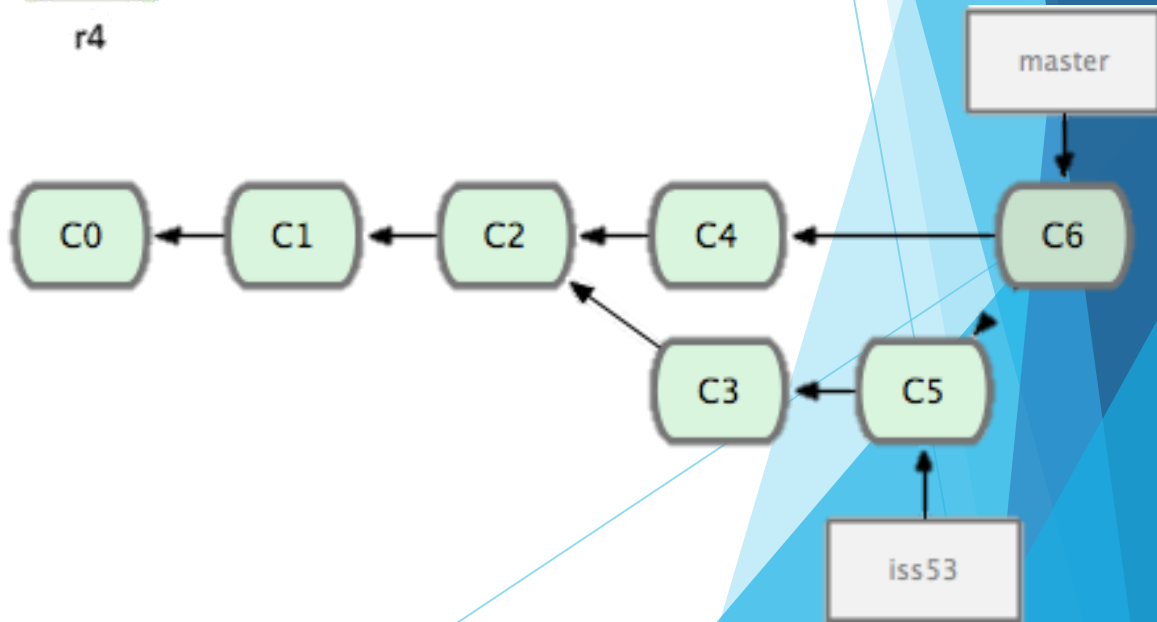
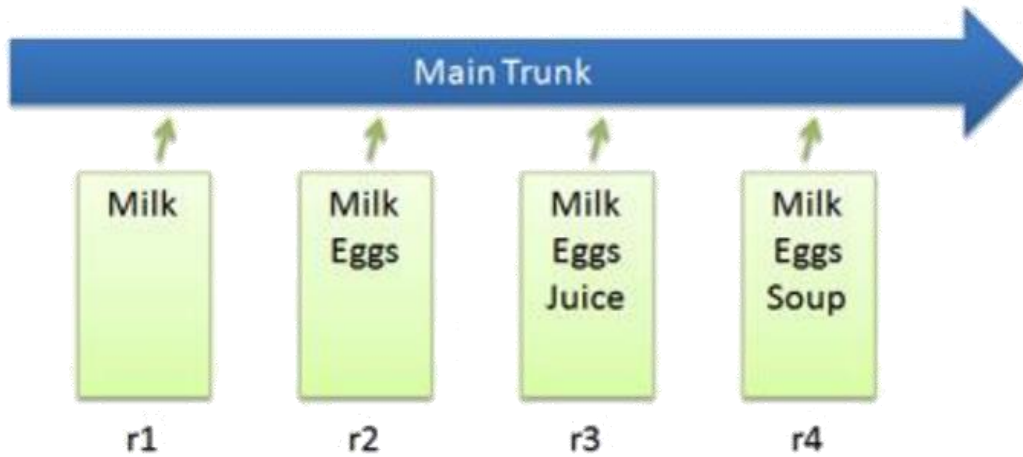
Терминология VCS 2

Метка	tag	именованная версия проекта
Ветка	branch	«метка, имеющая историю развития»
Коммит, фиксация	commit, checkin	сохранение изменений в репозитории
Чекаут, извлечение	checkout	получение рабочей копии (без истории)
Push, pull		отправка/получение изменений в/из другого репозитория
cvcs HEAD svn trunk git master		Основная ветка

Иллюстрация работы с VCS

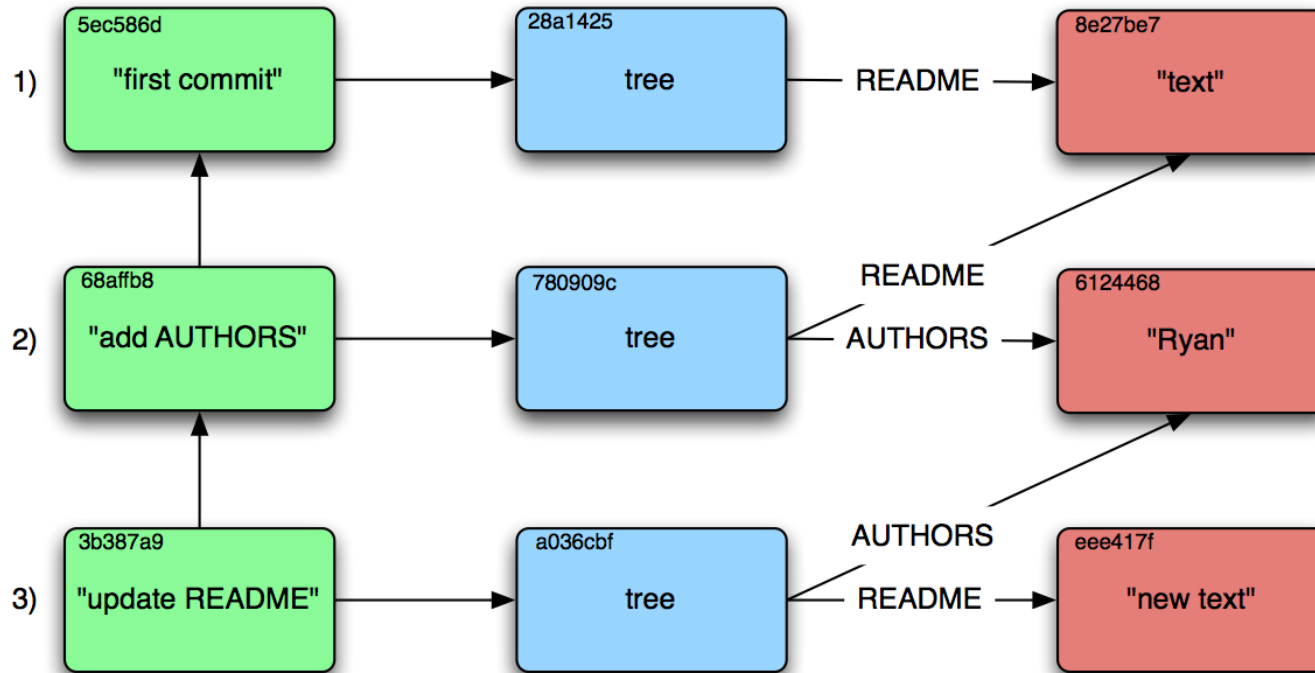
КОММИТЫ

- История неизменна
- Ничего не теряется



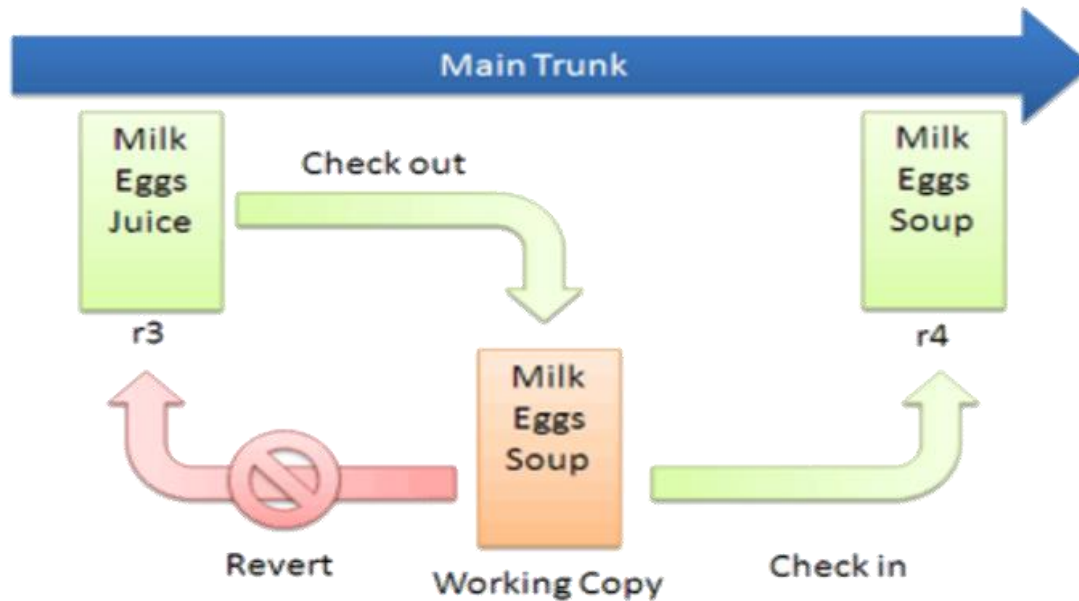
Изменения

- Хранятся не файлы целиком, а изменения



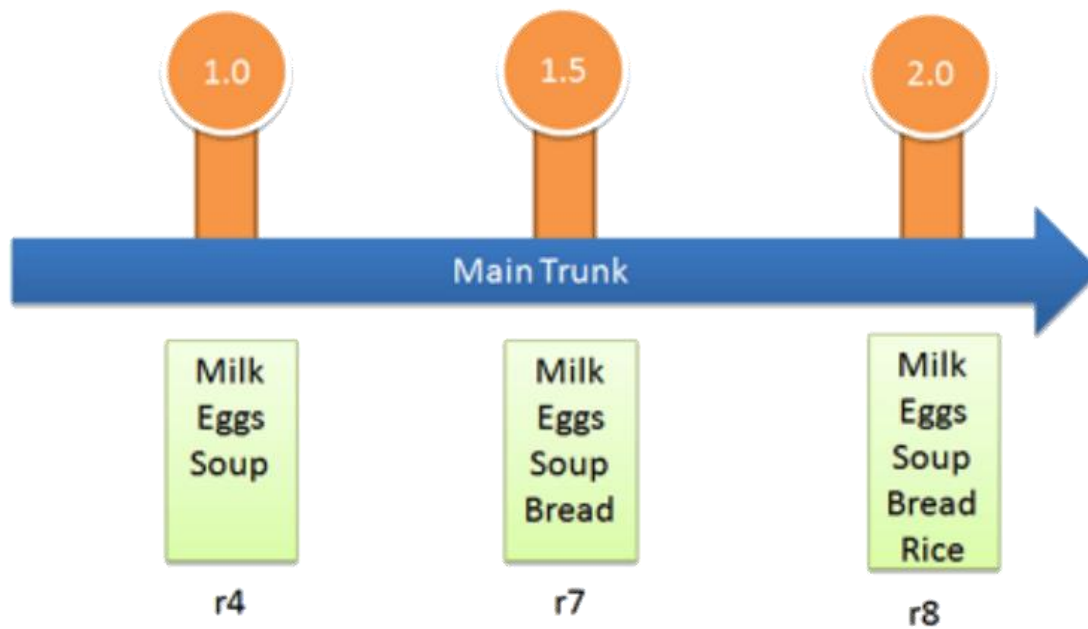
Редактирование

- Можем отменить внесенные изменения
- Можем получить любую предыдущую версию файла



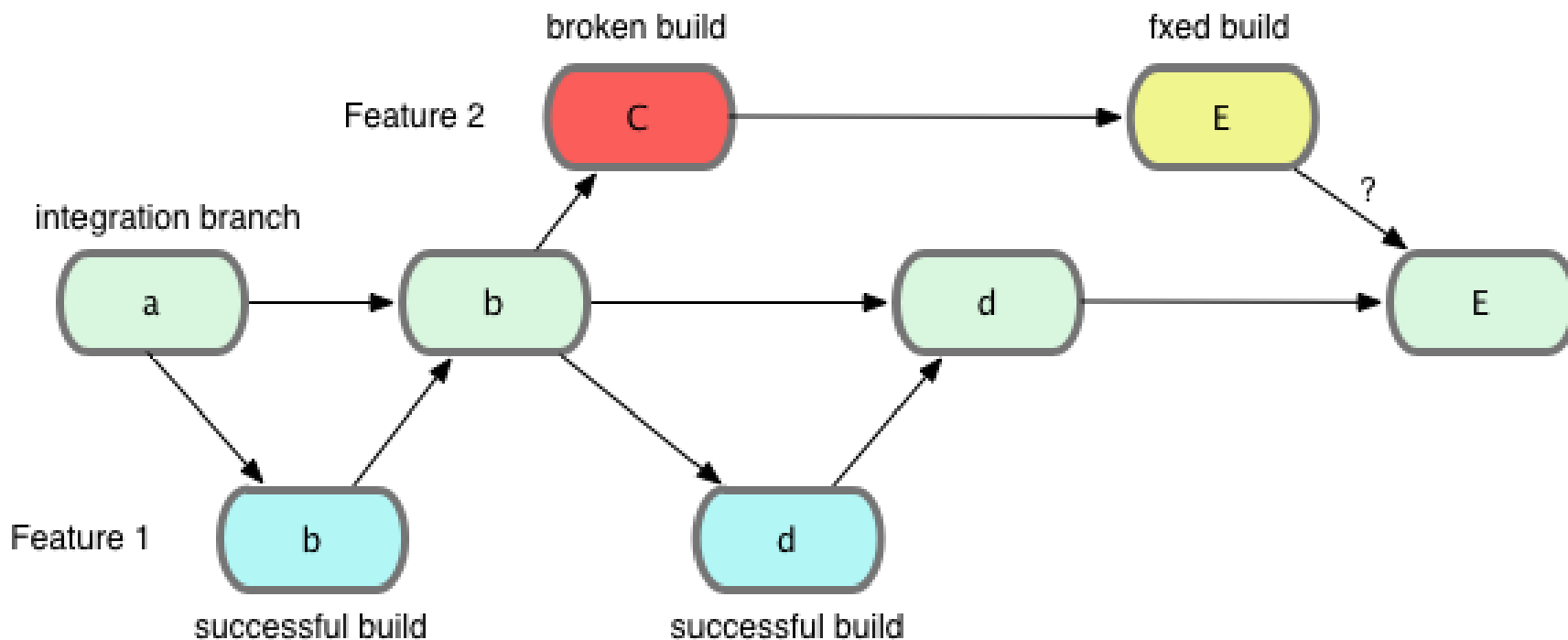
Тегирование

- Любому коммиту может быть назначена метка
- Метка включает в себя номер версии и краткое описание
- Коммит легко найти по его метке



Ветвление

- Не бойтесь делать много веток
- На каждую фичу - свою ветку!
- Ветка - указатель на последовательность КОММИТОВ

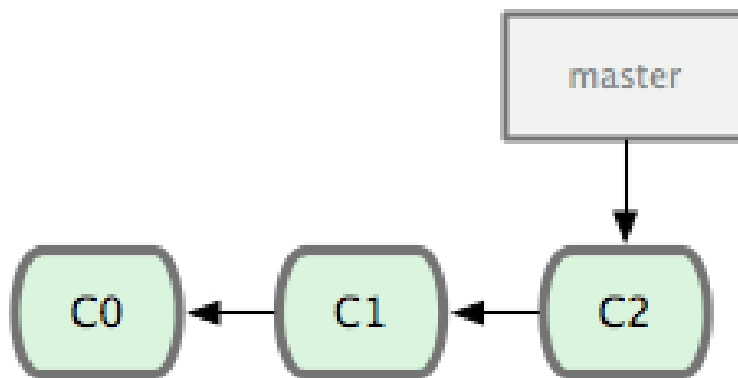


Ветвление и слияние

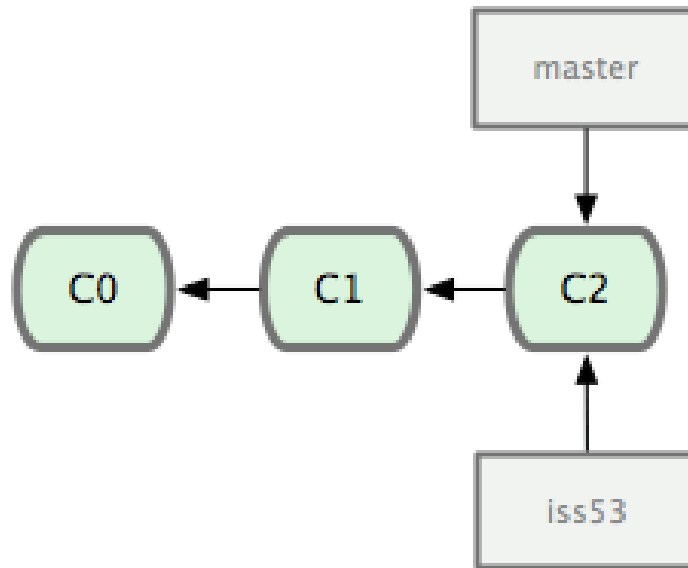
Терминология VCS 3

Слияние	merge	Объединение изменений из разных веток
Перестановка	rebase	Применение изменений из одной ветки на другую
Конфликт	conflict	Попытка объединить два по-разному измененных файла

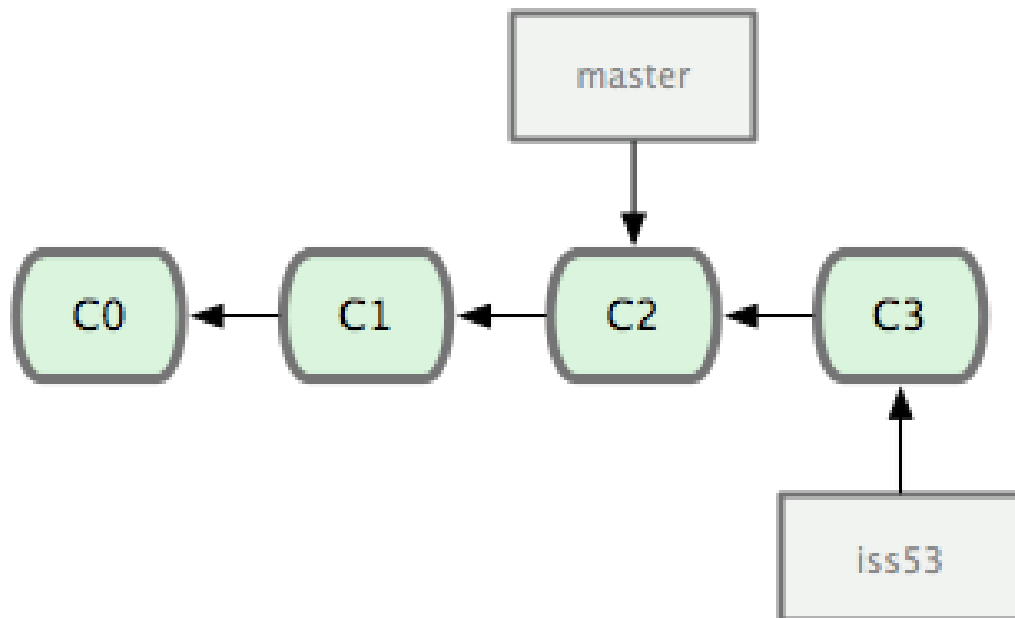
- ▶ Для начала представим, что вы работаете над своим проектом и уже имеете пару коммитов



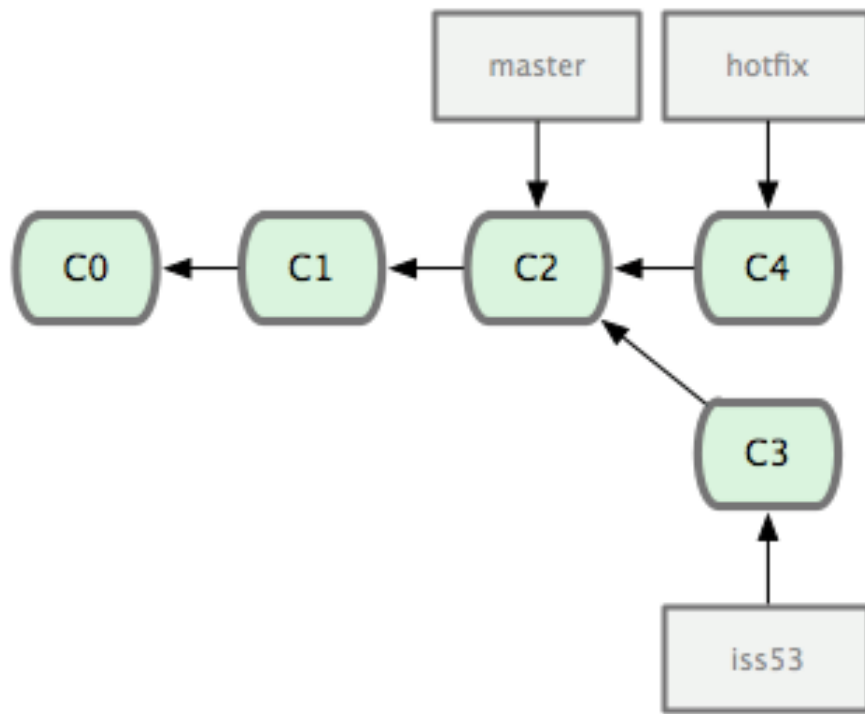
- ▶ А теперь Вы решили, что вы будете работать над проблемой №53 из системы отслеживания ошибок, используемой вашей компанией.
- ▶ Так как проблема №53 является обособленной задачей, над которой вы собираетесь работать, мы создадим новую ветку и будем работать на ней.



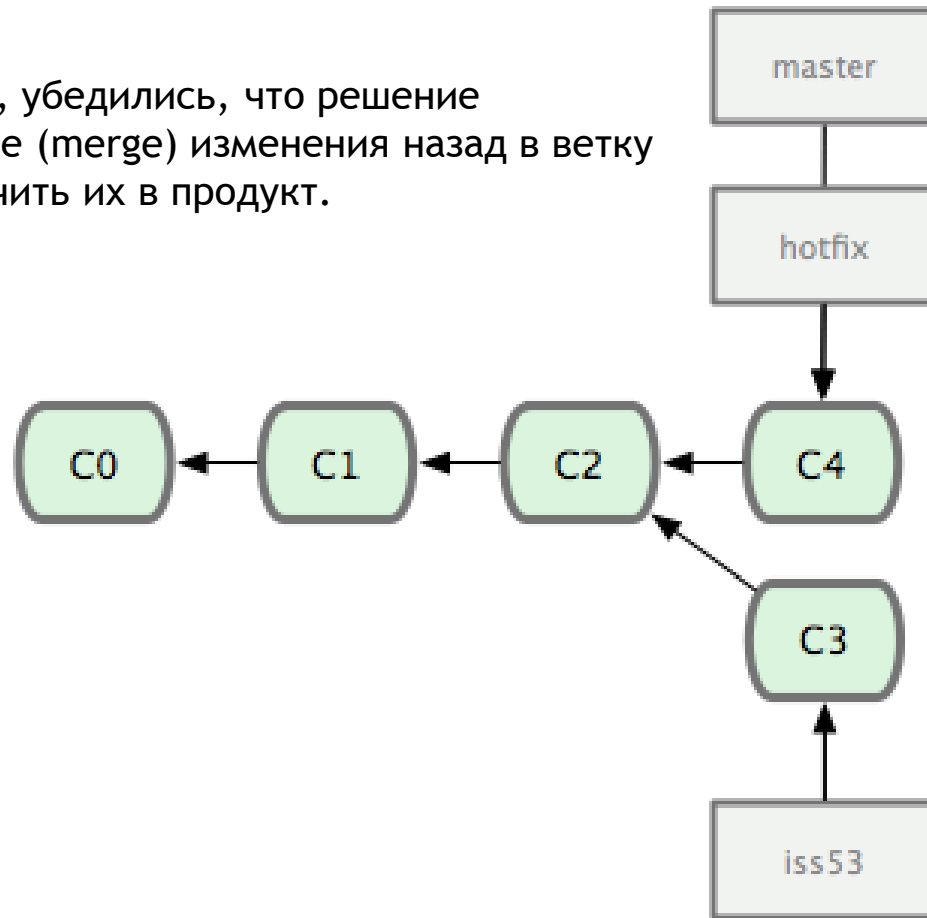
- ▶ Во время работы Вы делаете несколько коммитов.
- ▶ Эти действия сдвигают ветку `iss53` вперёд потому, что вы на неё перешли (то есть ваш HEAD указывает на неё)
- ▶ Master остаётся на месте



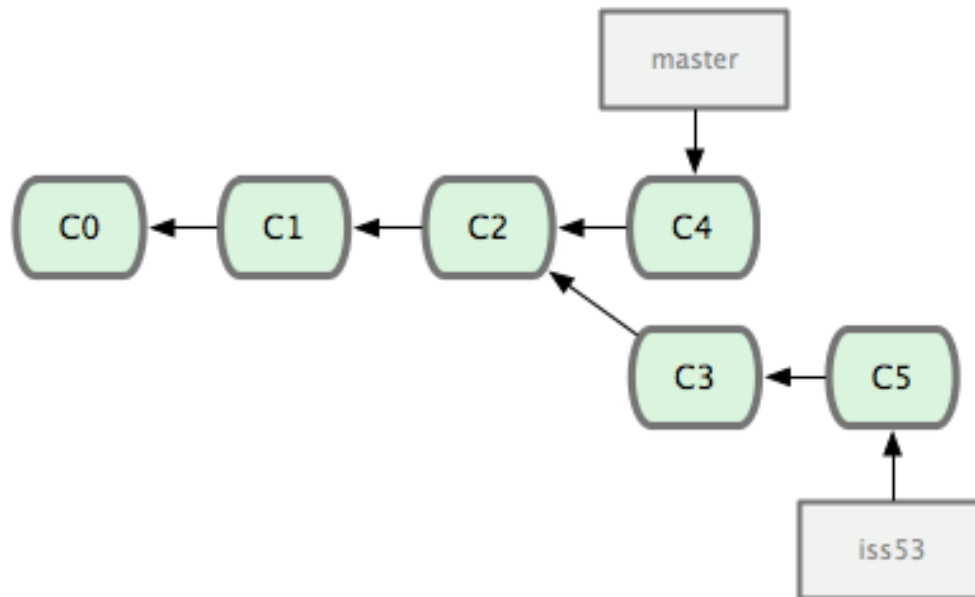
- ▶ Теперь Вы получаете звонок о том, что есть проблема, которую необходимо немедленно устранить.
- ▶ Нет нужды делать исправления поверх тех изменений, которые вы уже сделали в iss53.
- ▶ Всё, что вам нужно сделать, это перейти на ветку master.



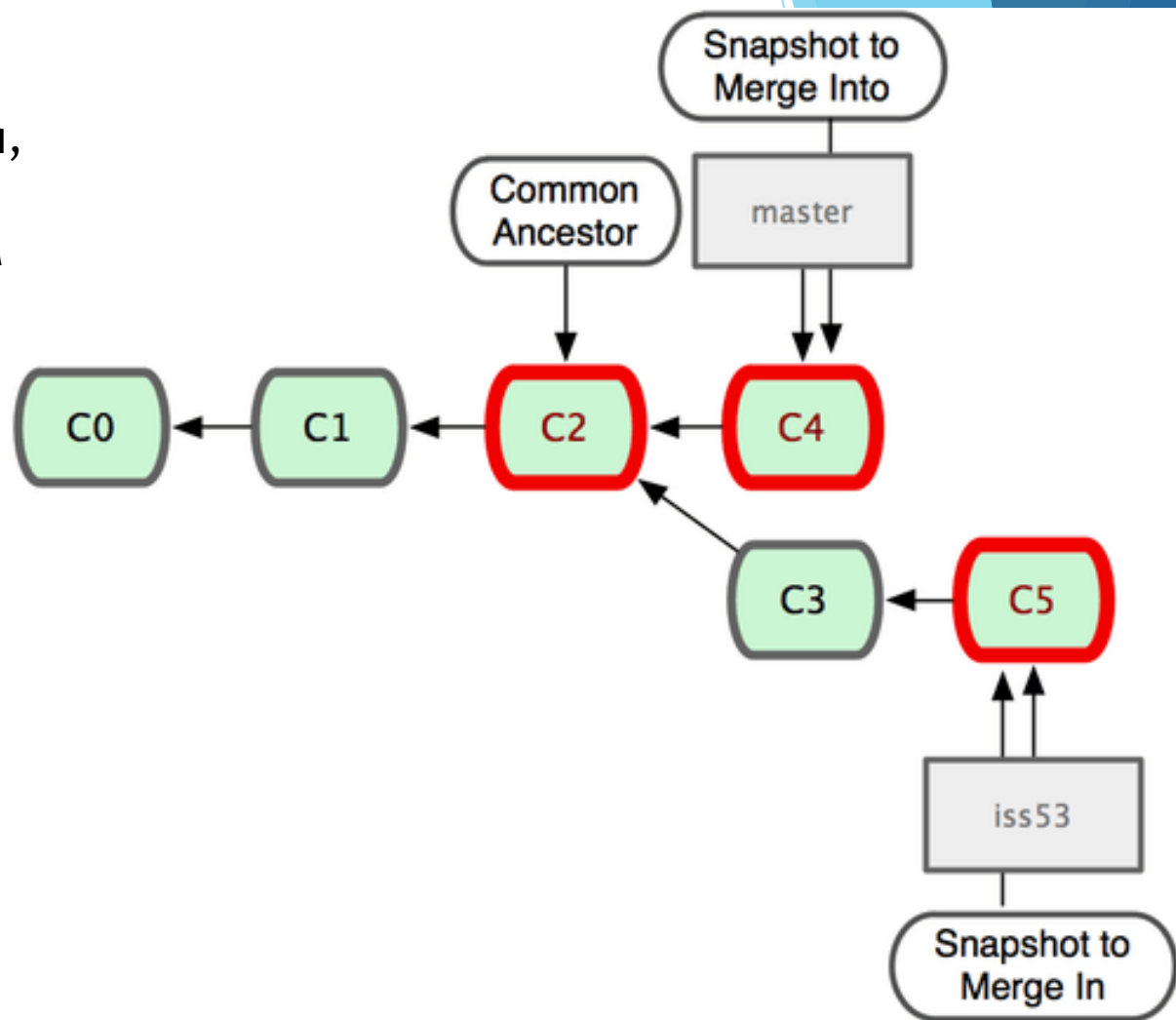
- ▶ Вы запустили тесты, убедились, что решение работает, и сливаете (merge) изменения назад в ветку master, чтобы включить их в продукт.



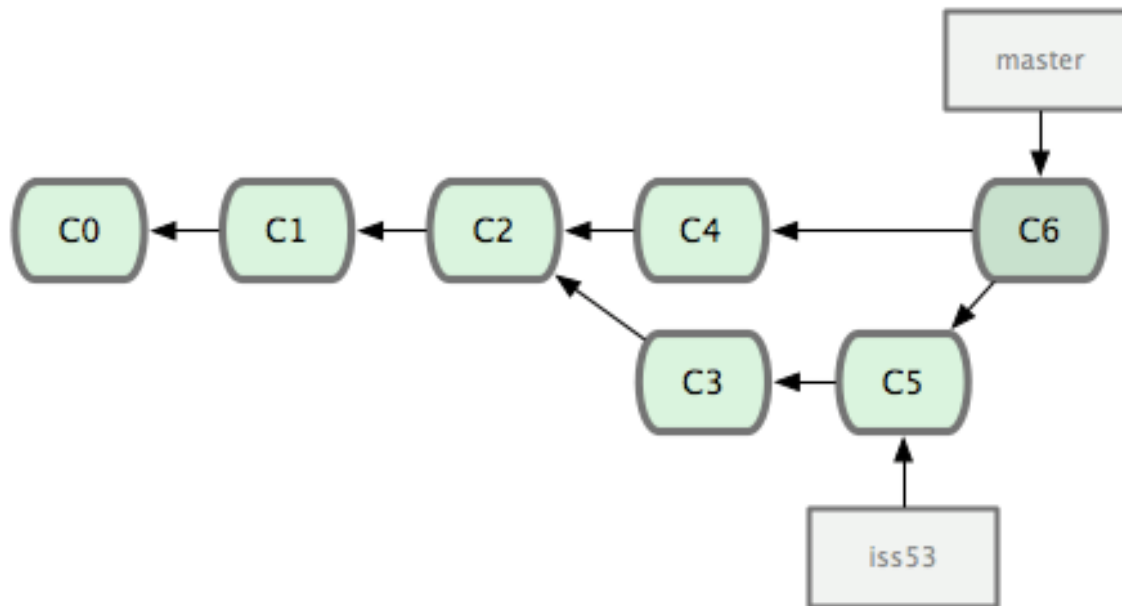
- ▶ Теперь вы можете вернуться обратно к рабочей ветке для проблемы №53 и продолжить работать над ней
- ▶ Допустим, вы разобрались с проблемой №53 и сделали коммит C5.
- ▶ Теперь вы готовы объединить эту ветку и свой master. Чтобы сделать это, мы сольём ветку iss53



- ▶ Это слияние немного отличается от слияния, сделанного ранее для ветки hotfix. В данном случае история разработки разделилась в некоторой точке.
- ▶ Так как коммит на той ветке, на которой вы находитесь, не является прямым предком для ветки, которую вы сливаете, VCS придётся проделать кое-какую работу



- ▶ VCS автоматически создаёт новый коммит, содержащий результаты слияния (при условии, что не будет конфликтов)



Основы конфликтов при слиянии

- ▶ Иногда процесс слияния не идёт гладко.
- ▶ Если вы изменили одну и ту же часть файла по-разному в двух ветках, которые собираетесь слить, VCS не сможет сделать это чисто.
- ▶ Например, если ваше решение проблемы №53 изменяет ту же часть файла, что и hotfix, вы получите **конфликт слияния**

```
$ git merge iss53 Auto-merging index.html CONFLICT  
(content): Merge conflict in index.html Automatic  
merge failed; fix conflicts and then commit the  
result.
```

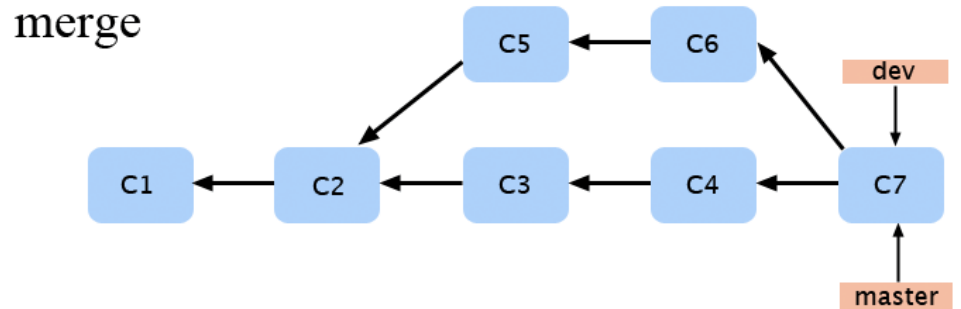
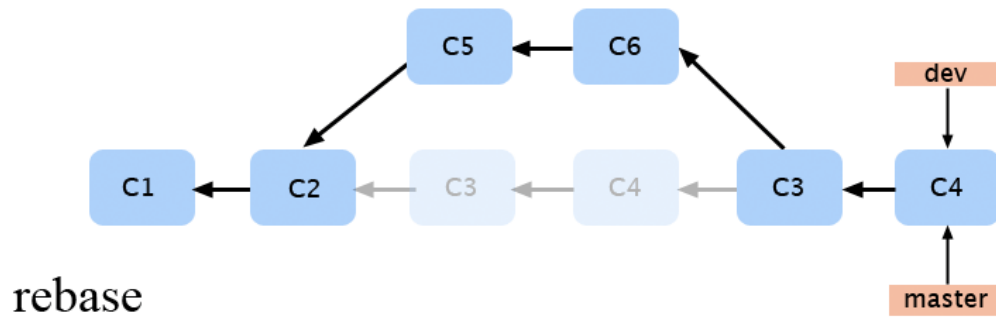
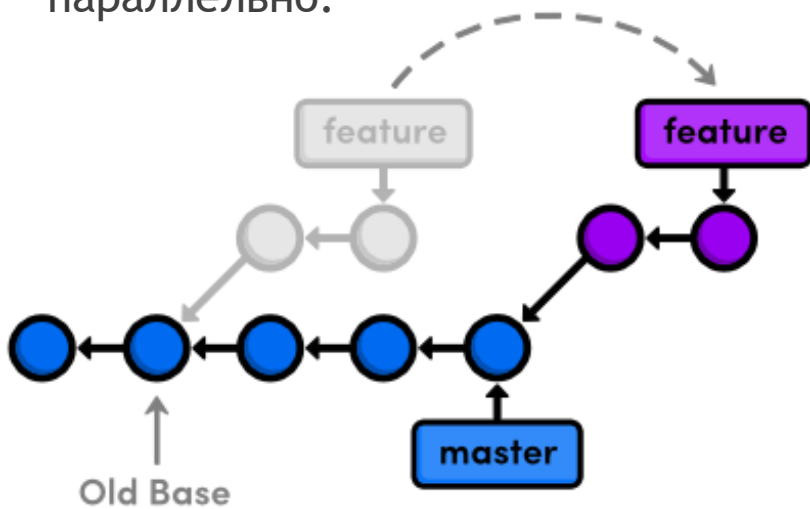
- ▶ VCS приостановит процесс слияния до тех пор, пока вы не разрешите конфликт
- ▶ Всё, что имеет отношение к конфликту слияния и что не было разрешено, отмечено как unmerged.

```
<<<<<<< HEAD:index.html
<div id="footer">contact :
email.support@github.com</div>
=====
<div id="footer"> please contact us at
support@github.com </div>
>>>>>> iss53:index.html
```

- ▶ После того, как Вы устраните все конфликты, можно выполнить коммит для завершения слияния.

Rebase

- ▶ Кроме слияния изменений (merge), некоторые системы контроля версия позволяют выполнять перестановку (rebase).
- ▶ Rebase позволяет добиться более наглядной истории изменений
- ▶ Кажется, что работа над проектом велась последовательно, хотя, на самом деле, велась параллельно.





ОСНОВЫ Git

План

- ▶ Что такое Git?
- ▶ Установка и настройка Git
- ▶ Репозиторий Git
- ▶ Работа с репозиторием
- ▶ Работа с branch
- ▶ Игнорирование файлов
- ▶ Github и Bitbucket
- ▶ UI для работы с git

Что такое Git

- ▶ Git - распределенная система контроля версий
- ▶ Первая версия появилась в 2005 г.
- ▶ Разработана Линусом Торвальдсом для использования в управлении разработкой ядра Linux.

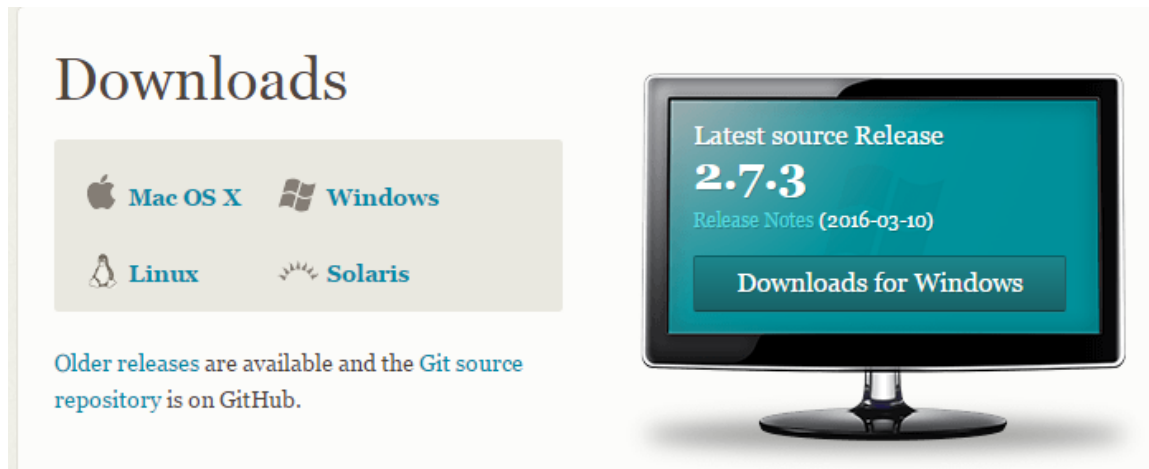


Ключевые особенности Git

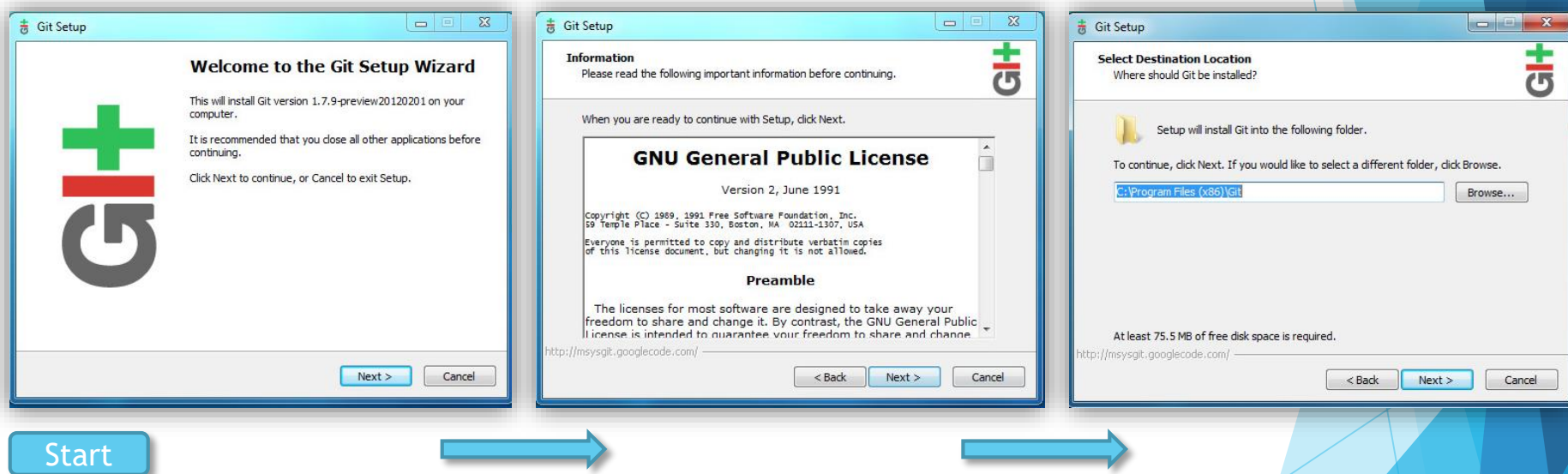
- ▶ Поддерживается автономная работа; локальные фиксации изменений могут быть отправлены позже.
- ▶ Каждое рабочее дерево в Git содержит хранилище с полной историей проекта.
- ▶ Ни одно хранилище Git не является по своей природе более важным, чем любое другое.
- ▶ Скорость работы, ветвление делается быстро и легко.

Установка и настройка

- ▶ Загрузить Git можно с официального сайта: <http://git-scm.com/>



- ▶ Установка выполняется как и обычной программы. Необходимо указать каталог для установки и некоторые параметры.



- ▶ Для минимальной настройки Git на компьютере необходимо задать глобальные параметры, которые будут применяться к вносимым изменениям и подписывать их. Они будут указывать на Вас в истории коммитов в удаленных репозиториях.
- ▶ Такими глобальными настройками являются имя пользователя и его email. Их можно установить следующими командами в консоли Git:

```
$ git config --global user.name "Your name"  
$ git config --global user.email email@example.com
```

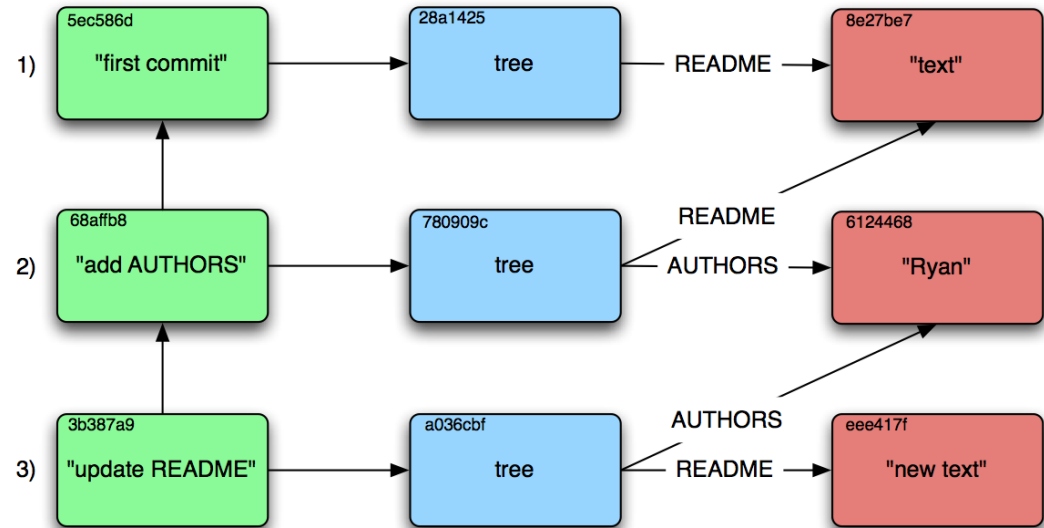
- ▶ Все параметры будут помещены в файл с настройками Git `.gitconfig`, расположенным в домашнем каталоге пользователя.

Репозиторий Git

- ▶ Git хранит информацию в структуре данных называемой репозиторий (repository).
- ▶ Репозиторий хранит:
 - Набор коммитов (commit objects)
 - Набор ссылок на коммиты (heads).
- ▶ Репозиторий хранится в той же директории, что и сам проект в поддиректории `.git`.

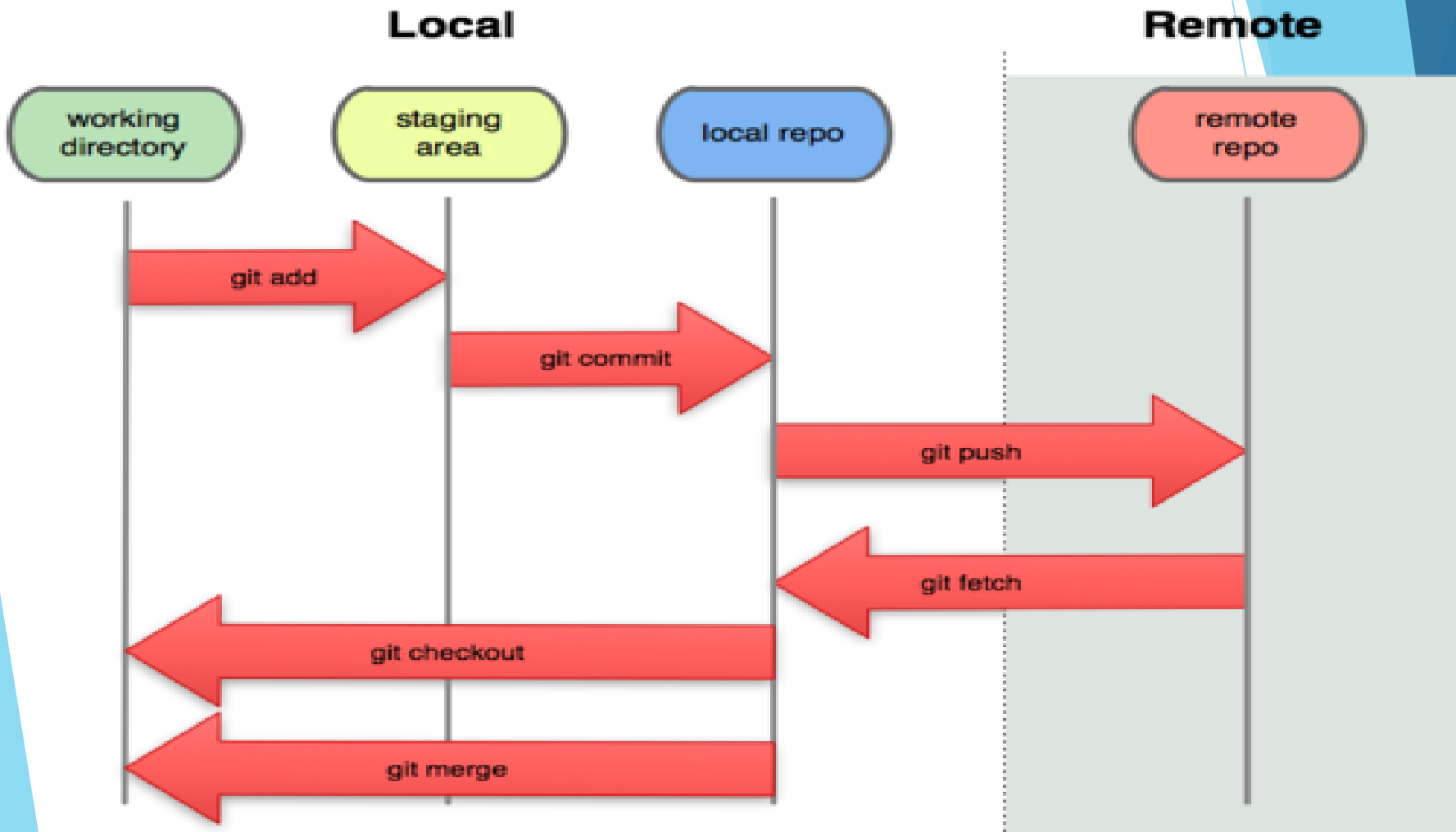
Commit objects содержат:

- ▶ Набор файлов, отображающий состояние проекта в текущую точку времени
- ▶ Ссылки на родительские commit objects
- ▶ SHA1 имя - 40 символьная строка которая уникально идентифицирует commit object.
Идентичные коммиты всегда будут иметь одинаковое имя.



Основные команды

- ▶ `git init` - создание репозитория
- ▶ `git add <имена файлов>` - Добавляет файлы в индекс
- ▶ `git commit` - выполняет коммит проиндексированных файлов в репозиторий
- ▶ `git status` - показывает какие файлы изменились между текущей стадией и HEAD. Файлы разделяются на 3 категории: новые файлы, измененные файлы, добавленные новые файлы
- ▶ `git checkout <sha1 или метка>` - получение указанной версии файла
- ▶ `git push` - отправка изменений в удаленный репозиторий
- ▶ `git fetch` - получение изменений из удаленного репозитория
- ▶ `git clone <remote url>` - клонирование удаленного репозитория себе



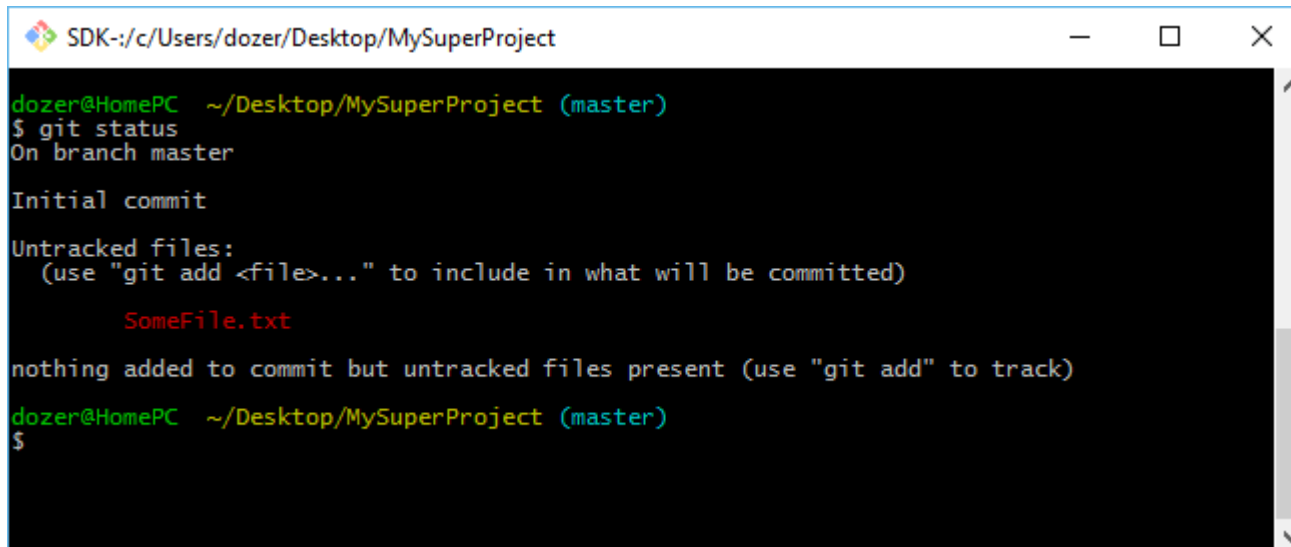
Пример: создание репозитория

- ▶ Сначала перейдем в папку, в которой хотим создать новый репозиторий git
- ▶ Воспользуемся командой `git init`
- ▶ Появилась новая скрытая папка: `.git` - наш репозиторий
- ▶ Кроме того, была создана ветка `master`

```
SDK-:/c/Users/dozer/Desktop/MySuperProject
dozer@HomePC ~/Desktop/MySuperProject
$ cd "C:\Users\dozer\Desktop\MySuperProject"
dozer@HomePC ~/Desktop/MySuperProject
$ git init
Initialized empty Git repository in C:/Users/dozer/Desktop/MySuperProject/.git/
dozer@HomePC ~/Desktop/MySuperProject (master)
$ ls -A
.git/
dozer@HomePC ~/Desktop/MySuperProject (master)
$
```

Пример: добавление файлов и создание КОММИТОВ

- ▶ Создадим в нашей папке новый файл. Пусть это будет текстовый документ `SomeFile.txt` с парой строк внутри.
- ▶ Используем команду `git status`. Git сообщает нам, что появился новый файл, который пока не добавлен в индекс



```
SDK-:/c/Users/dozer/Desktop/MySuperProject

dozer@HomePC ~/Desktop/MySuperProject (master)
$ git status
On branch master

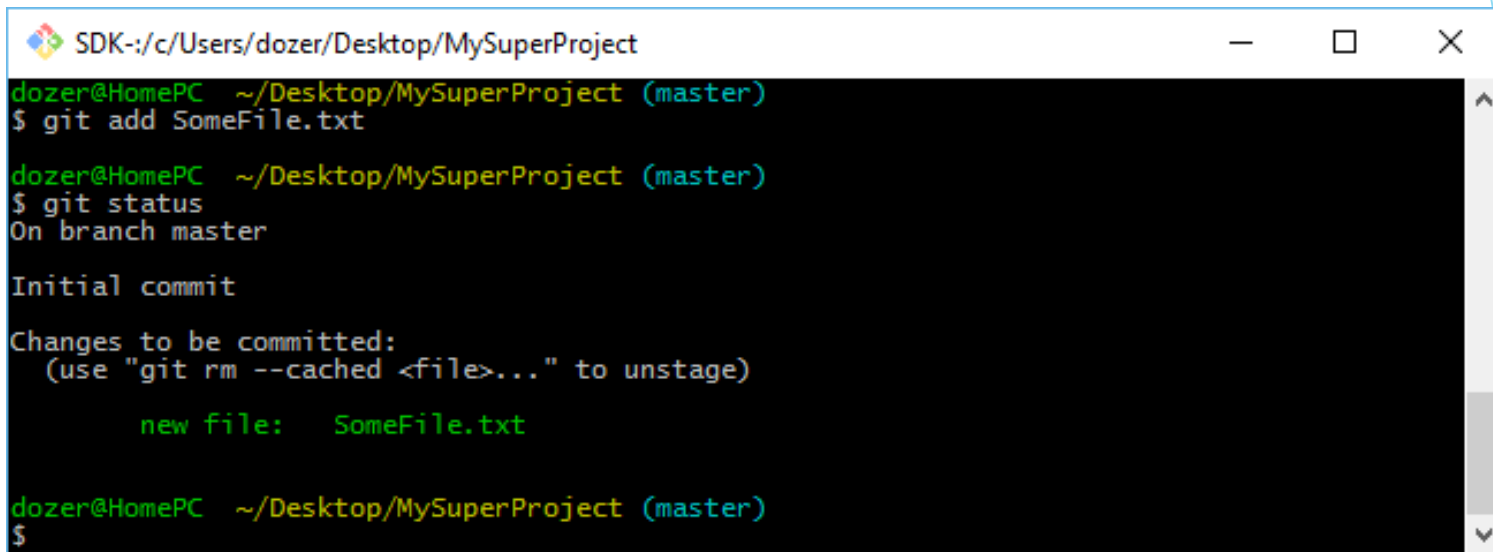
Initial commit

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

       SomeFile.txt

nothing added to commit but untracked files present (use "git add" to track)
dozer@HomePC ~/Desktop/MySuperProject (master)
$
```

- ▶ Давайте добавим наш текстовый файл в систему контроля версий.
- ▶ Прописываем `git add SomeFile.txt`
- ▶ Если ещё раз запросить статус (`git status`), мы увидим, что теперь наш файл подсвечен зеленым и добавлен в индекс

A screenshot of a terminal window titled "SDK-:/c/Users/dozer/Desktop/MySuperProject". The terminal shows the following sequence of commands and output:

```
dozer@HomePC ~/Desktop/MySuperProject (master)
$ git add SomeFile.txt

dozer@HomePC ~/Desktop/MySuperProject (master)
$ git status
On branch master

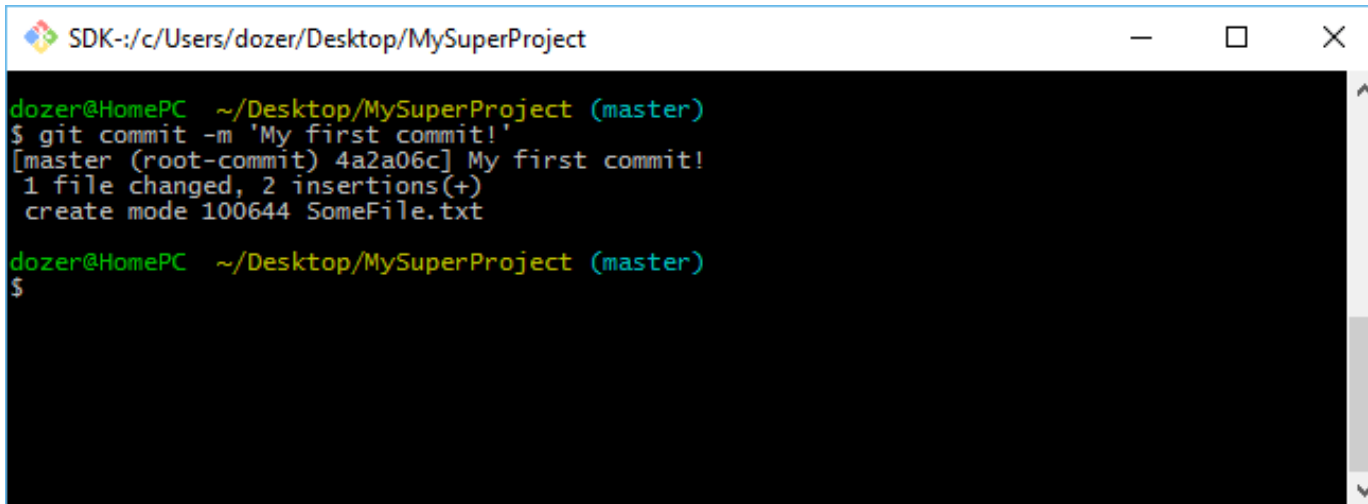
Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   SomeFile.txt

dozer@HomePC ~/Desktop/MySuperProject (master)
$
```

- ▶ Нам осталось только зафиксировать изменения, создав **новый коммит**.
- ▶ Для этого воспользуемся командой **git commit** с ключом **-m** (добавить информационное сообщение к коммиту)
- ▶ Git проинформирует нас об успешном создании **нового коммита**



```
SDK-:/c/Users/dozer/Desktop/MySuperProject
dozer@HomePC ~/Desktop/MySuperProject (master)
$ git commit -m 'My first commit!'
[master (root-commit) 4a2a06c] My first commit!
1 file changed, 2 insertions(+)
 create mode 100644 SomeFile.txt

dozer@HomePC ~/Desktop/MySuperProject (master)
$
```

Полезные команды

- ▶ `git log` - показывает лог commits начиная с HEAD
- ▶ `git remote` - показывает информацию об удаленных репозиториях, а также позволяет удалять и добавлять их
- ▶ `git mv` - используется для перемещения или переименования файла
- ▶ `git rm` - удаляет файл из репозитория не затрагивая рабочую копию
- ▶ `gitk` - визуальная утилита для работы с репозиторием

Работа с ветками

- ▶ Создание branch выполняется следующей командой:

```
git branch branch_name
```

- ▶ Переключение веток осуществляется командой:

```
git checkout branch_name
```

- ▶ Данная команда выполняет 2 функции:

- ▶ Сдвинет указатель HEAD на `branch_name`
- ▶ Перезапишет все файлы в директории на соответствующие новому HEAD

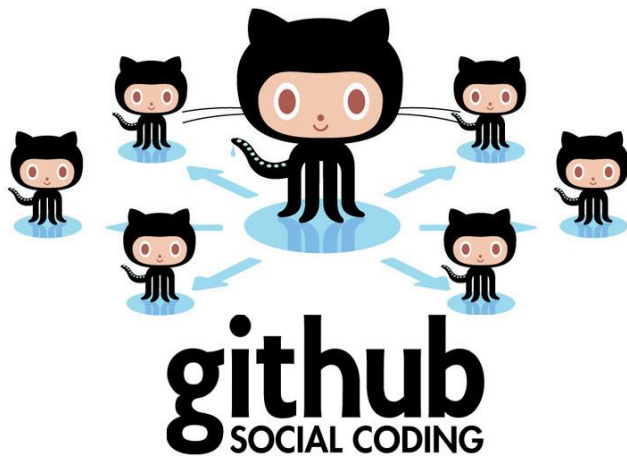
Игнорирование файлов

- ▶ Далеко не все файлы проекта требуется включать в систему контроля версий:
 - ▶ Результаты сборки
 - ▶ Настройки IDE
 - ▶ Файлы кэша
 - ▶ Индивидуальные файлы пользователя
- ▶ Для этого используется файл **.gitignore**

- .gitignore - обычный текстовый файл
- Можно использовать регулярные выражения и подстановки
- Можно вставлять комментарии

```
1  ## Ignore Visual Studio temporary files, build results, and
2  ## files generated by popular Visual Studio add-ons.
3
4  # User-specific files
5  *.suo
6  *.user
7  *.userosscache
8  *.sln.docstates
9
10 # User-specific files (MonoDevelop/Xamarin Studio)
11 *.userprefs
12
13 # Build results
14 [Dd]ebug/
15 [Dd]ebugPublic/
16 [Rr]elease/
17 [Rr]eleases/
18 x64/
19 x86/
20 bld/
21 [Bb]in/
22 [Oo]bj/
23 [Ll]og/
..
```


Github и Bitbucket



GitHub

- ▶ GitHub – крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git.
- ▶ Создатели сайта называют GitHub «социальной сетью для разработчиков»



Особенности

- ▶ Сервис **абсолютно бесплатен** для проектов с открытым исходным кодом и предоставляет им все возможности. (для частных проектов предлагаются различные платные тарифные планы)
- ▶ Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.
- ▶ Программисты могут объединять свои репозитории – GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.
- ▶ Для проектов есть личные страницы, вики-разметка и система отслеживания ошибок.
- ▶ Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

➤ GitHub - социальная сеть для разработчиков!

The screenshot shows the GitHub profile of Pavel Egorov xoposhiy. The profile includes a profile picture, a bio, location (Ekaterinburg), and a link to a VK profile. It also displays statistics: 19 followers, 44 starred repositories, and 1 following. The 'Organizations' section shows a logo for SKB Kontur. The main content area is divided into three sections: 'Popular repositories', 'Repositories contributed to', and 'Public contributions'. The 'Public contributions' section features a heatmap showing activity from March to February, with a summary of 432 total contributions in the last year, an 8-day longest streak, and a 0-day current streak.

Search GitHub

Pull requests Issues Gist

Contributions Repositories Public activity Unfollow

Popular repositories

cvk2012 Аналитика по политическому компасу проек...	5 ★
banality Банальности	2 ★
loopbreakers	2 ★
creeps AI for screeps.com	2 ★
edo.interconnection Разработка формата для роуминга операто...	1 ★

Repositories contributed to

kontur-edu/uLearn E-learning platform and courses	2 ★
kontur-intern-2... /internship-20... Студенческая выездная школа-интенсив по...	0 ★
kontur-kampus... /kontur-kampus Материалы со школы Контур.Кампус 2015 в...	1 ★
ChrisPenner/Advent-Of-Code-P... Examples of "Advent Of Code" solutions in ma...	42 ★
mr146/masterspaper Master's paper	1 ★

Public contributions

Summary of pull requests, issues opened, and commits. [Learn how we count contributions.](#) Less More

Contributions in the last year 432 total Mar 10, 2015 – Mar 10, 2016	Longest streak 8 days August 7 – August 14	Current streak 0 days Last contributed 23 hours ago
---	---	--

Bitbucket

- ▶ Bitbucket – веб-сервис для хостинга проектов и их совместной разработки, основанный на системе контроля версий Mercurial и Git. По назначению и предлагаемым функциям аналогичен GitHub.
- ▶ Для публичных репозиториев количество пользователей не ограничено (*BitBucket* бесплатен для проектов открытого программного обеспечения).
- ▶ К частному (закрытому) репозиторию может иметь доступ до пяти пользователей; большее количество записей предоставляется в рамках платного обслуживания.



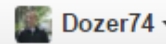
Пример: работа с GitHub

- ▶ Давайте отправим ранее созданный репозиторий на github.
- ▶ Для начала необходимо создать новый репозиторий на сайте: <https://github.com/new>
- ▶ Заполняем имя проекта. Заодно можем создать файлы readme и .gitignore (github предоставляет шаблоны игнор файлов для большинства языков программирования)

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Dozer74

Repository name

MySuperProject



Great repository names are short and memorable. Need inspiration? How about **upgrad**

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're re

Add .gitignore: None

Add a license: None



Create repository

- ▶ Github подсказывает, какие команды необходимо использовать, чтобы отправить наши файлы на сервер:

...or push an existing repository from the command line

```
git remote add origin https://github.com/Dozer74/MySuperProject.git
git push -u origin master
```

- ▶ Теперь заглянем на страницу нашего проекта:

No description or website provided. — Edit

1 commit 1 branch 0 releases 1 contributor

Branch: master **New pull request** New file Upload files Find file HTTPS https://github.com/Dozer74 Download ZIP

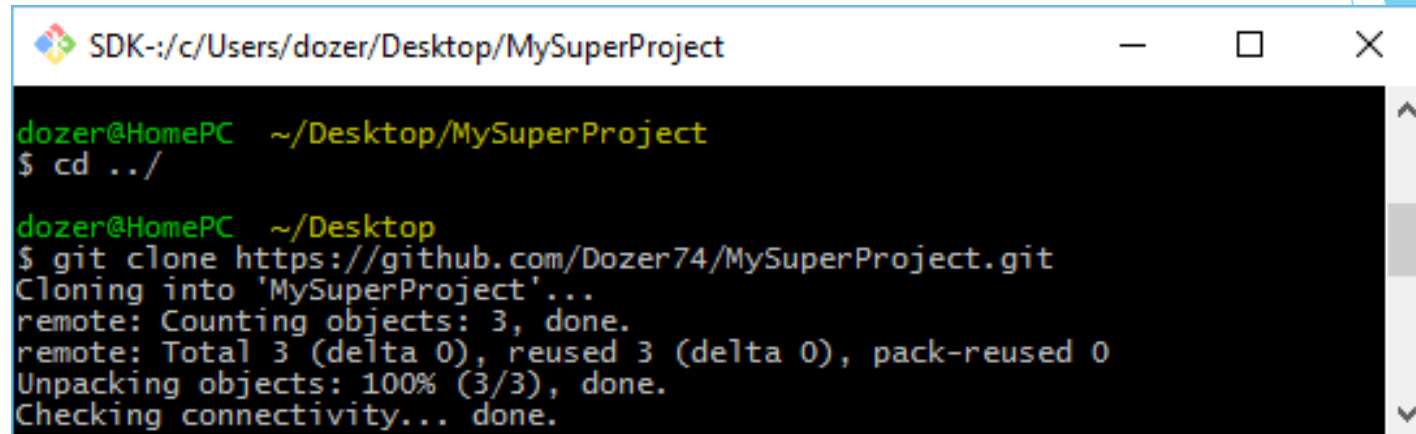
Dozer74 My first commit! Latest commit 4a2a06c 17 minutes ago

SomeFile.txt My first commit! 17 minutes ago

Help people interested in this repository understand your project by adding a README. **Add a README**

- ▶ Репозиторий успешно отправлен на удаленный сервер!

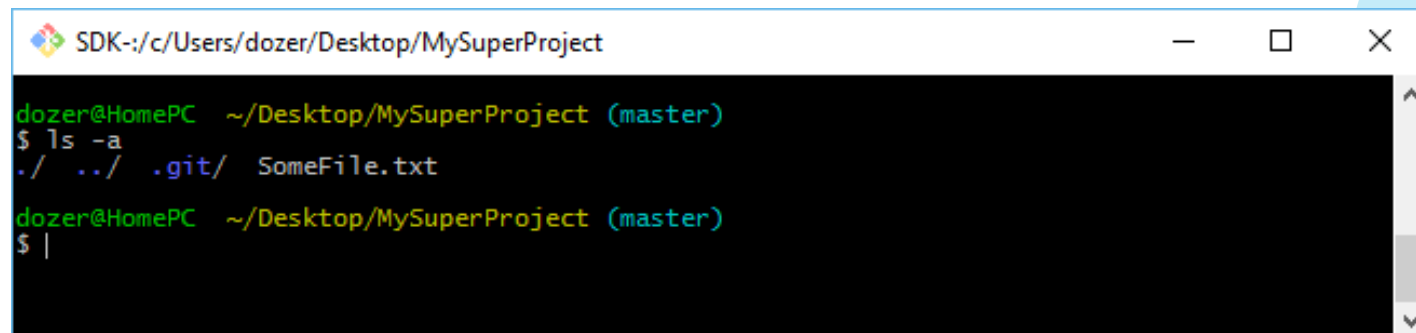
- ▶ Теперь рассмотрим процесс клонирования уже существующего репозитория
- ▶ Удалим наш локальный репозиторий, а затем выполним команду `git clone <url>` (url репозитория можно посмотреть на его странице на сайте github)



```
SDK-:/c/Users/dozer/Desktop/MySuperProject
dozer@HomePC ~/Desktop/MySuperProject
$ cd ../

dozer@HomePC ~/Desktop
$ git clone https://github.com/Dozer74/MySuperProject.git
Cloning into 'MySuperProject'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

- ▶ Мы получили копию нашего репозитория и все файлы из него

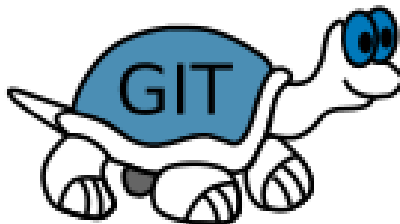


```
SDK-:/c/Users/dozer/Desktop/MySuperProject
dozer@HomePC ~/Desktop/MySuperProject (master)
$ ls -a
./ ../ .git/ SomeFile.txt

dozer@HomePC ~/Desktop/MySuperProject (master)
$ |
```

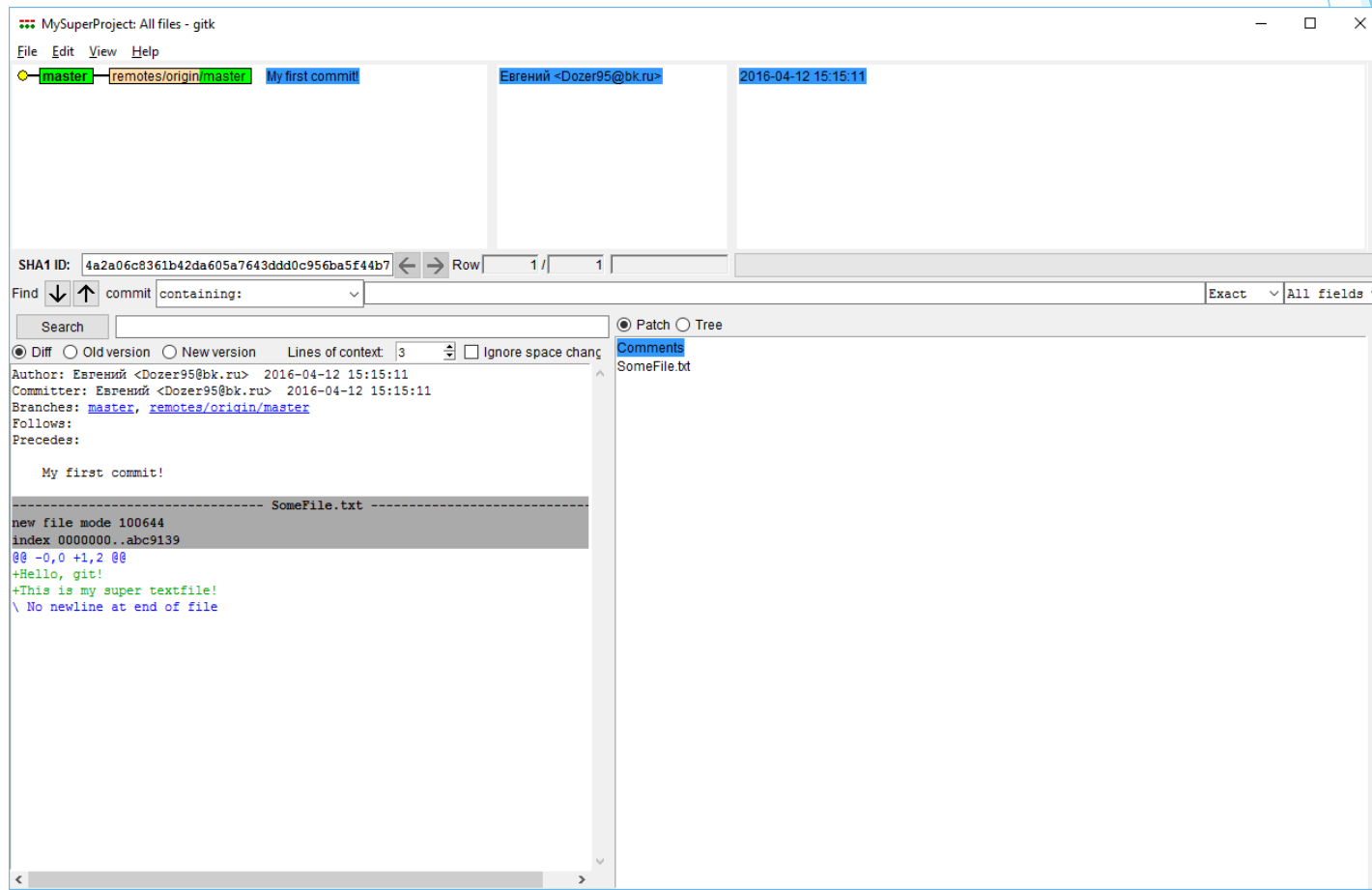

UI для работы с Git

- ▶ Не всем нравится использовать консольные команды
- ▶ Для работы git существует множество визуальных оболочек и программ. Вот лишь некоторые из них:
 - ▶ Gitk
 - ▶ Github Desktop
 - ▶ TortoiseGit
 - ▶ Atlassian SourceTree



Gitk

- ▶ Поставляется вместе с git
- ▶ Простая визуальная утилита, позволяющая просматривать текущий репозиторий, историю коммитов и изменений в нем.



The screenshot shows the Gitk application window titled "MySuperProject: All files - gitk". The interface includes a menu bar (File, Edit, View, Help) and a header area with commit information: "master", "remotes/origin/master", "My first commit", "Евгений <Dozer95@bk.ru>", and "2016-04-12 15:15:11". Below this is a search bar with "SHA1 ID: 4a2a06c8361b42da605a7643ddd0c956ba5f44b7" and a "Find" dropdown set to "commit containing:". The main area displays commit metadata and a diff for "SomeFile.txt".

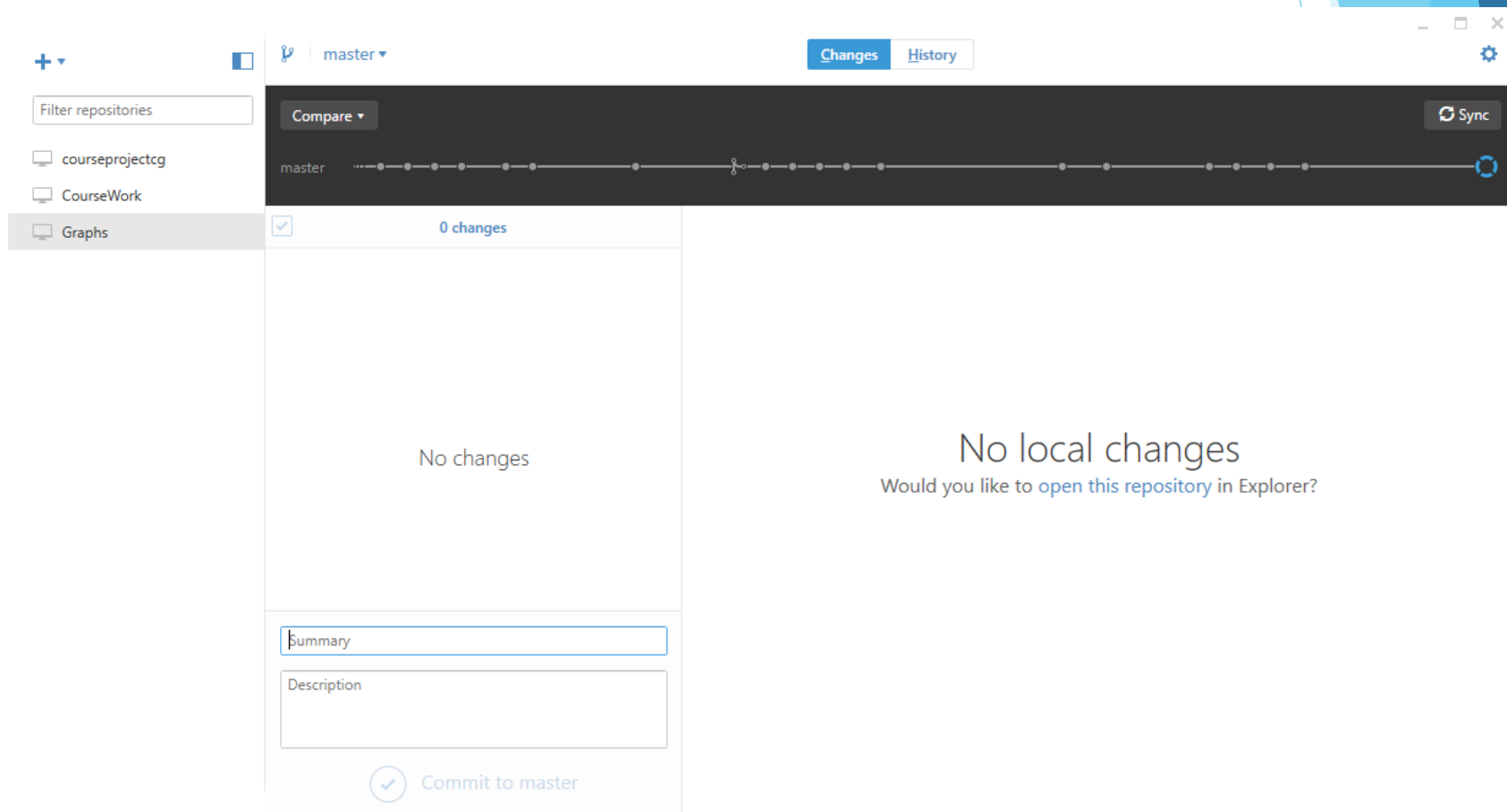
```
Author: Евгений <Dozer95@bk.ru> 2016-04-12 15:15:11
Committer: Евгений <Dozer95@bk.ru> 2016-04-12 15:15:11
Branches: master, remotes/origin/master
Follows:
Precedes:

My first commit!

----- SomeFile.txt -----
new file mode 100644
index 0000000..abc9139
@@ -0,0 +1,2 @@
+Hello, git!
+This is my super textfile!
\ No newline at end of file
```

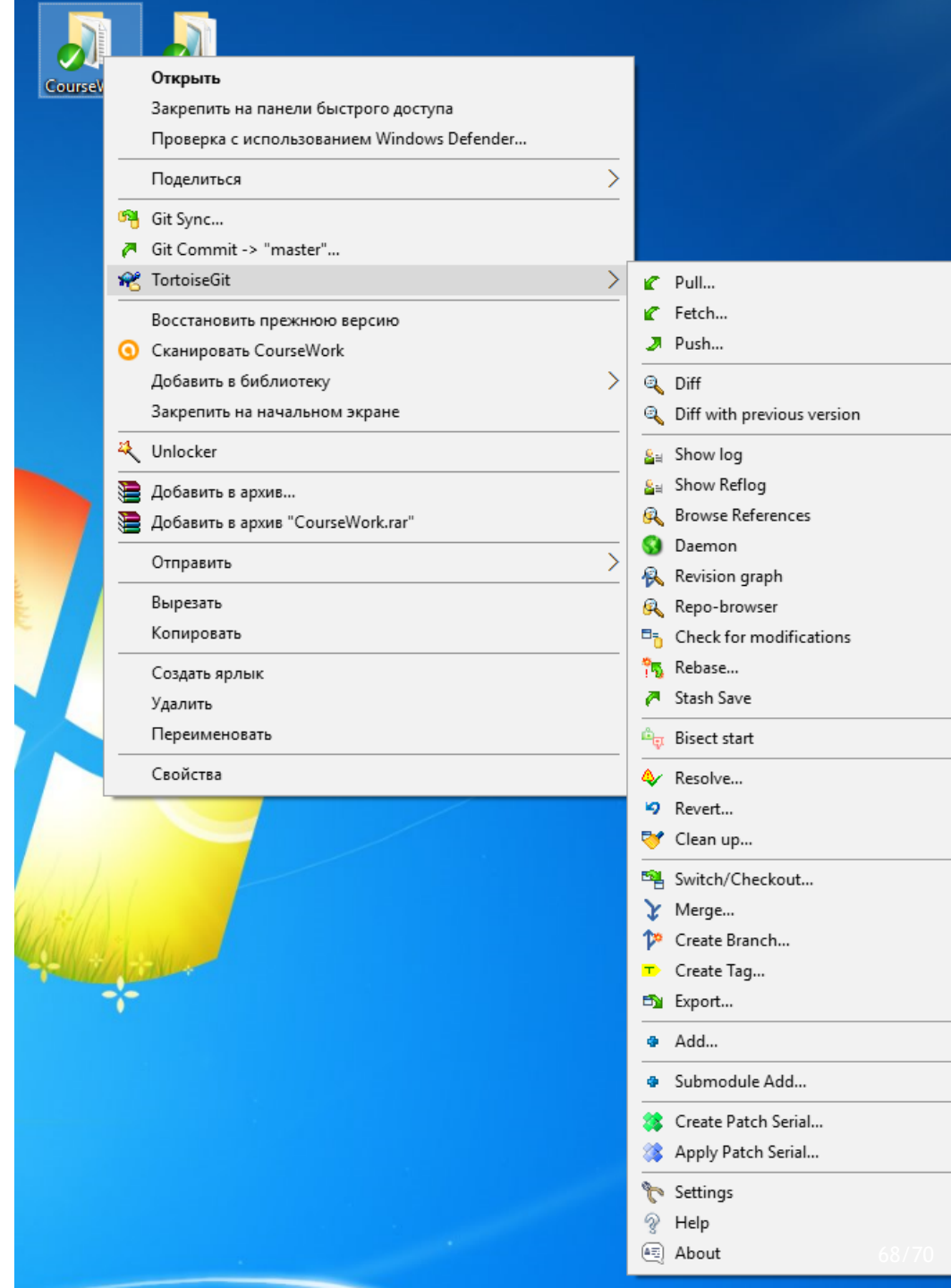
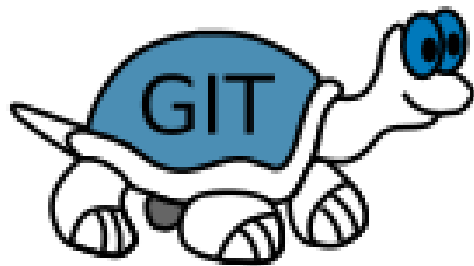
Github Desktop

- ▶ Разработана для удобной работы с github.
- ▶ Красивый интерфейс, мало возможностей
- ▶ Для большинства действий всё равно придется открывать КОНСОЛЬ :(



TortoiseGit

- ▶ «Черепашка» добавляет возможность работы с git в контекстное меню проводника
- ▶ Большинство функций доступны в несколько щелчков мыши
- ▶ Не очень наглядна, требует больше времени на освоение



Atlassian SourceTree

- ▶ Разработана создателями Bitbucket
- ▶ Похожа на Github Desktop, позволяет получить доступ к большому числу возможностей git

The screenshot shows the SourceTree application window titled "sourcetree-website (Git)". The interface includes a toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, and Stash. Below the toolbar, there are dropdown menus for "All Branches", "Show Remote Branches", and "Ancestor Order", along with a "Jump to:" field. The main area is divided into a left sidebar and a main table.

Workspace Sidebar:

- WORKSPACE
- File status
- History
- Search
- BRANCHES
 - css_buttons
 - hero_images
 - master**
 - retina
- TAGS
- REMOTES
 - origin
- STASHES
- SUBMODULES
- SUBTREES

Commit History Table:

Graph	Commit	Author	Description	Date
	b7358c7	Rahul Chha...	master origin/master origin/HEAD Removing ol...	Mar 3, 2016, 11:...
	bdb8bef	Rahul Chhab...	Merged in update-google-verification (pull request #14)	Feb 18, 2016, 1:3...
	dfe975d	Tyler Tadej...	origin/update-google-verification Update google verificati...	Feb 11, 2016, 2:2...
	3bc3290	Tyler Tadej...	Replace outdated Atlassian logo in footer with base-64 en...	Feb 11, 2016, 2:1...
	dba47f9	Tyler Tadej...	Add gitignore	Feb 11, 2016, 1:3...
	ff67b45	Mike Minns...	Updated Mac min-spec to 10.10	Feb 15, 2016, 11:...
	72d32a8	Michael Min...	Merged in hero_images (pull request #13)	Feb 15, 2016, 10:...
	246c4ff	Joel Unger...	origin/hero_images hero_images Used Tinypng to c...	Feb 11, 2016, 3:3...
	9d9438c	Joel Unger...	Replacing hero images with new version of SourceTree	Feb 9, 2016, 2:59...
	ce75b63	Michael Min...	Merged in bug/date-https (pull request #12)	Feb 15, 2016, 10:...
	85367bb	Patrick Tho...	origin/bug/date-https fixed date and https errors	Jan 7, 2016, 12:2...
	4f9b557	Joel Unger...	New Favicon	Feb 8, 2016, 3:55...
	384e6d5	Rahul Chhab...	origin/search-console-access search console google ver...	Feb 3, 2016, 2:09...
	6fa47a9	Mike Minns...	updated to move supported version to OSX 10.9+	Dec 15, 2015, 2:0...
	8dd87bb	Mike Minns...	remove extra , when a line is skipped due to empty server	Nov 23, 2015, 2:2...
	faa195e	Mike Minns...	Skip records with empty server/user id as gas rejects them	Nov 23, 2015, 2:1...
	0cdf96	Mike Minns...	corrected paths after merge	Nov 23, 2015, 2:0...
	051ab1b	Mike Minns...	corrected column counting	Nov 23, 2015, 1:5...
	a723bc2	Mike Minns...	Merge branch 'au2gex'	Nov 23, 2015, 1:5...
	65fd580	Mike Minns...	deal with invalid instanceids	Nov 23, 2015, 1:5...
	500a892	Michael Min...	Merged in au2gex (pull request #11)	Nov 23, 2015, 1:0...

Спасибо за внимание!