

# Системы контроля версий

# Системы контроля версий

- Системы управления версиями (Version Control Systems, VCS) или Системы управления исходным кодом (Source Management Systems, SMS) — важный аспект разработки современного ПО.
- VCS предоставляет следующие возможности:
  - Поддержка хранения файлов в репозитории.
  - Поддержка истории версий файлов в репозитории.
  - Нахождение конфликтов при изменении исходного кода и обеспечение синхронизации при работе в многопользовательской среде разработки.
  - Отслеживание авторов изменений.

# Классификация :

- Централизованные/распределённые — в централизованных системах контроля версий вся работа производится с центральным репозиторием, в распределённых — у каждого разработчика есть локальная копия репозитория.
- Блокирующие/не блокирующие — блокирующие системы контроля версий позволяют наложить запрет на изменение файла, пока один из разработчиков работает над ним, в неблокирующих один файл может одновременно изменяться несколькими разработчиками.
- Для текстовых данных/для бинарных данных — для VCS для текстовых данных очень важна поддержка слияния изменений, для VCS с бинарными данными важна возможность блокировки.

# Классификация :

- Централизованные/распределённые — в централизованных системах контроля версий вся работа производится с центральным репозиторием, в распределённых — у каждого разработчика есть локальная копия репозитория.
- Блокирующие/не блокирующие — блокирующие системы контроля версий позволяют наложить запрет на изменение файла, пока один из разработчиков работает над ним, в неблокирующих один файл может одновременно изменяться несколькими разработчиками.
- Для текстовых данных/для бинарных данных — для VCS для текстовых данных очень важна поддержка слияния изменений, для VCS с бинарными данными важна возможность блокировки.

# Ежедневный цикл работы

Обычный цикл работы разработчика выглядит следующим образом:

## **1. Обновление рабочей копии.**

Разработчик выполняет операцию обновления рабочей копии (update) насколько возможно

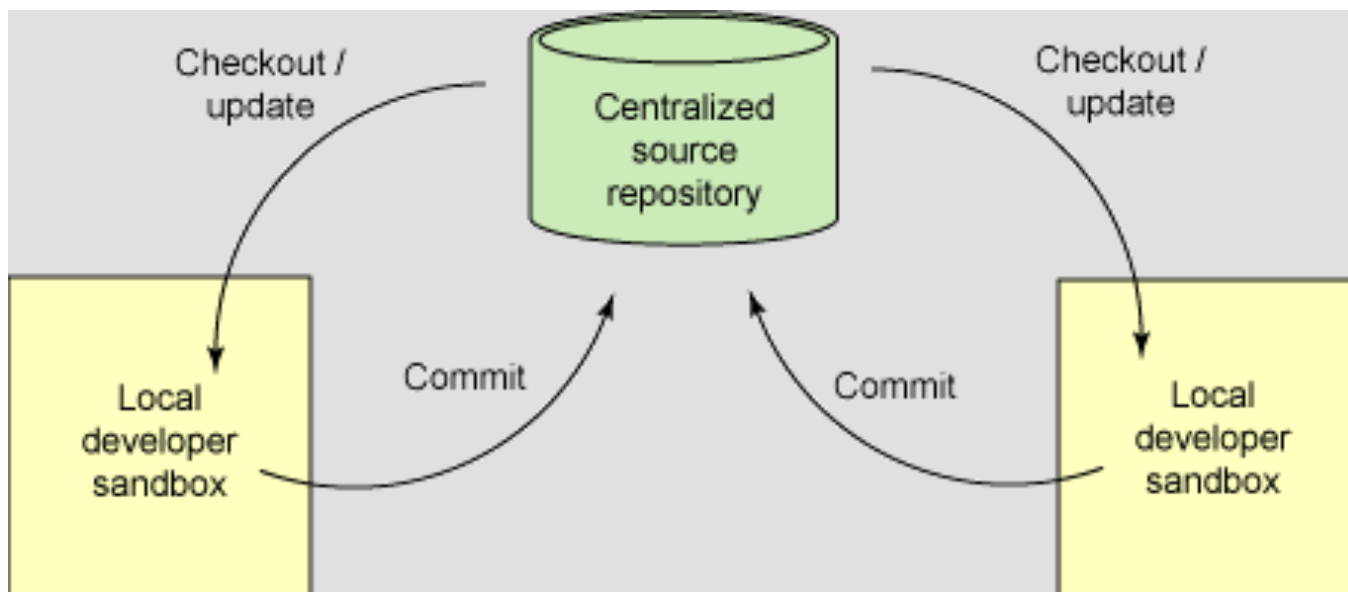
## **2. Модификация проекта.**

Разработчик локально модифицирует проект, изменяя входящие в него файлы в рабочей копии.

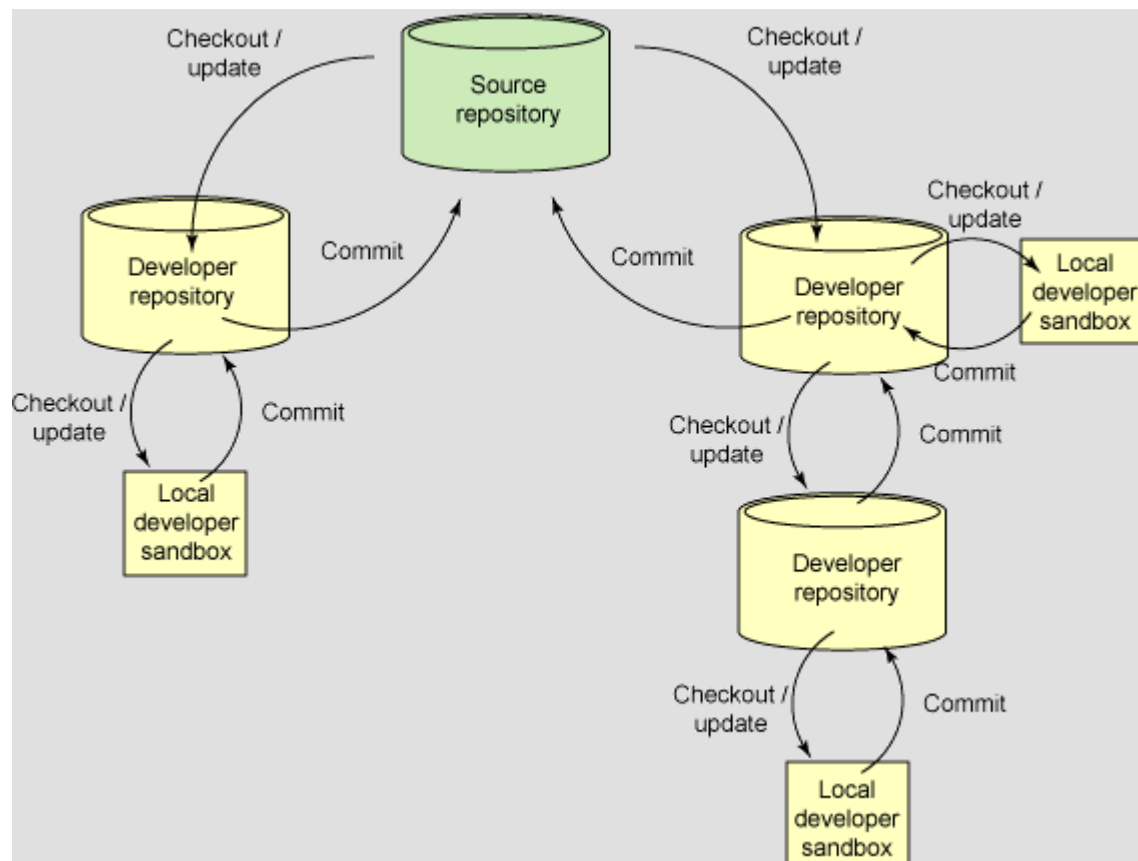
## **3. Фиксация изменений.**

Завершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер. VCS может требовать от разработчика перед фиксацией выполнить обновление.

# Централизованные VCS



# Распределенные VCS



# Основные термины

***working copy*** — рабочая (локальная) копия документов.

***repository, depot*** — хранилище.

***revision*** — версия документа. Новые изменения (changeset) создают новую ревизию репозитория.

***check-in, commit, submit*** — фиксация изменений.

***check-out, clone*** — извлечение документа из хранилища и создание рабочей копии.

***update, sync*** — синхронизация рабочей копии до некоторого заданного состояния хранилища (в т.ч. и к более старому состоянию, чем текущее).

***merge, integration*** — слияние независимых изменений.

***conflict*** — ситуация, когда несколько пользователей сделали изменения одного и того же участка документа.

***head*** — самая свежая версия (revision) в хранилище.

***origin*** — имя главного сервера



# Ветвление

**Ветвь (*branch*)** — направление разработки проекта, независимое от других.

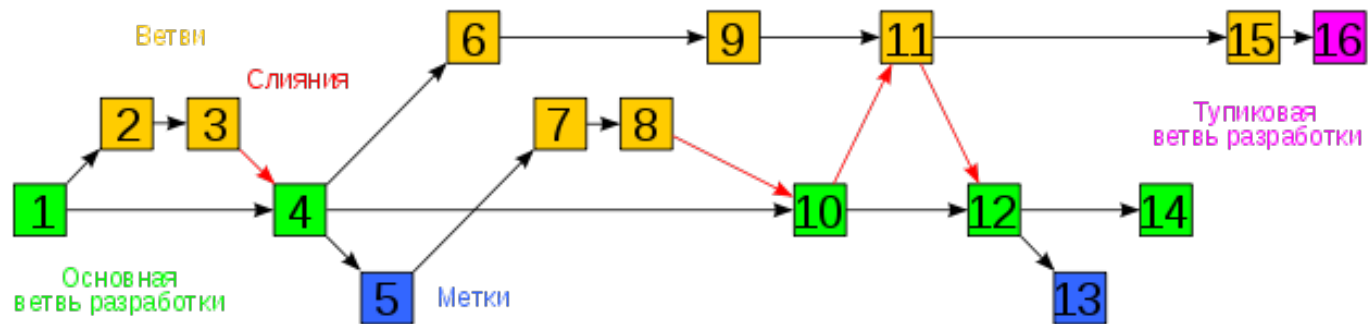
Ветвь представляет собой копию части (как правило, одного каталога) хранилища, в которую можно вносить свои изменения, не влияющие на другие ветви.

Документы в разных ветвях имеют одинаковую историю до точки ветвления и разные — после неё.

Изменения из одной ветви можно переносить в другую.

**Ствол (*trunk, mainline, master*)** — основная ветвь разработки проекта.

# Пример ветвления в проекте



# CVS

Одна из наиболее старых систем контроля версий. Создана в 1984 году как развитие RCS (Revision Control System), которая не поддерживала совместную работу.

Недостатки:

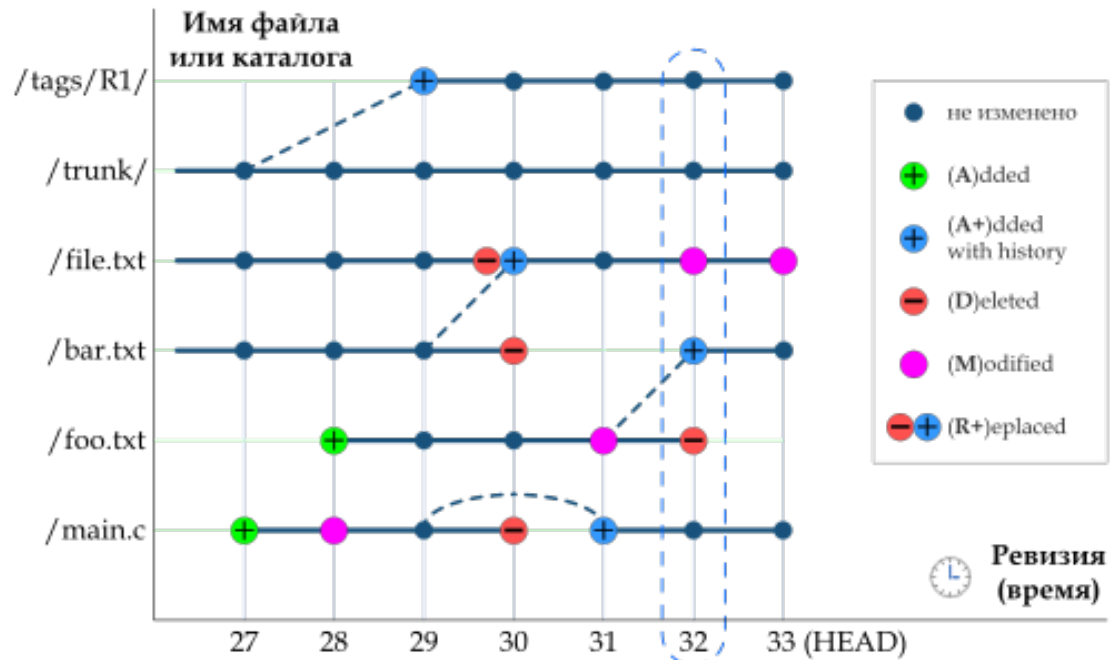
- Невозможно переименовать файл или директорию так, чтобы это изменение было отражено в истории.
- Ограниченная поддержка юникода и не-ASCII имен.
- Публикации изменений не атомарны.
- Наборы изменений не поддерживаются.
- Неэффективное хранение бинарных файлов.
- Оригинальный GNU CVS не поддерживает разграничения прав между пользователями репозитория.

# Subversion, SVN

Subversion, SVN — свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet Inc.

- Копирование объектов с разветвлением истории.
- Поддержка ветвления: создания ветвей (копированием директорий) и слияние ветвей (переносом изменений)
- Поддержка меток (копированием директорий).
- Поддержка разделение прав пользователей.
- История изменений и копии объектов (в том числе ветви и метки) хранятся в виде связанных разностных копий.
- Атомарная фиксации изменений в хранилище.

# Subversion, SVN



# Пример работы SVN

```
$ svn co http://projects.com/svn/myproject -username user  
$ cd myproject
```

..change project..

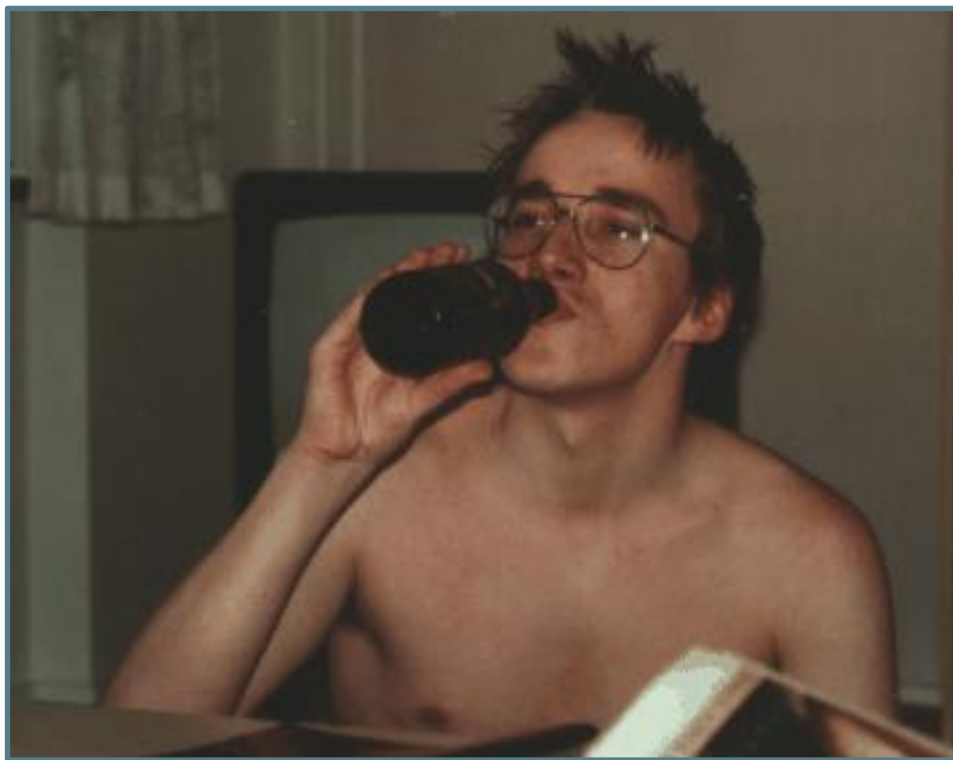
```
$ svn log  
$ svn st  
$ svn add newfile.cpp  
$ svn rm oldfile.cpp  
$ svn commit -m"Mega enhancement"
```

..next day..

```
$ svn up
```

# GIT

Помните этого парня?



# GIT

В 2005 году он решил записать свою VCS с простыми мерджами и децентрализованной структурой.

Достоинства:

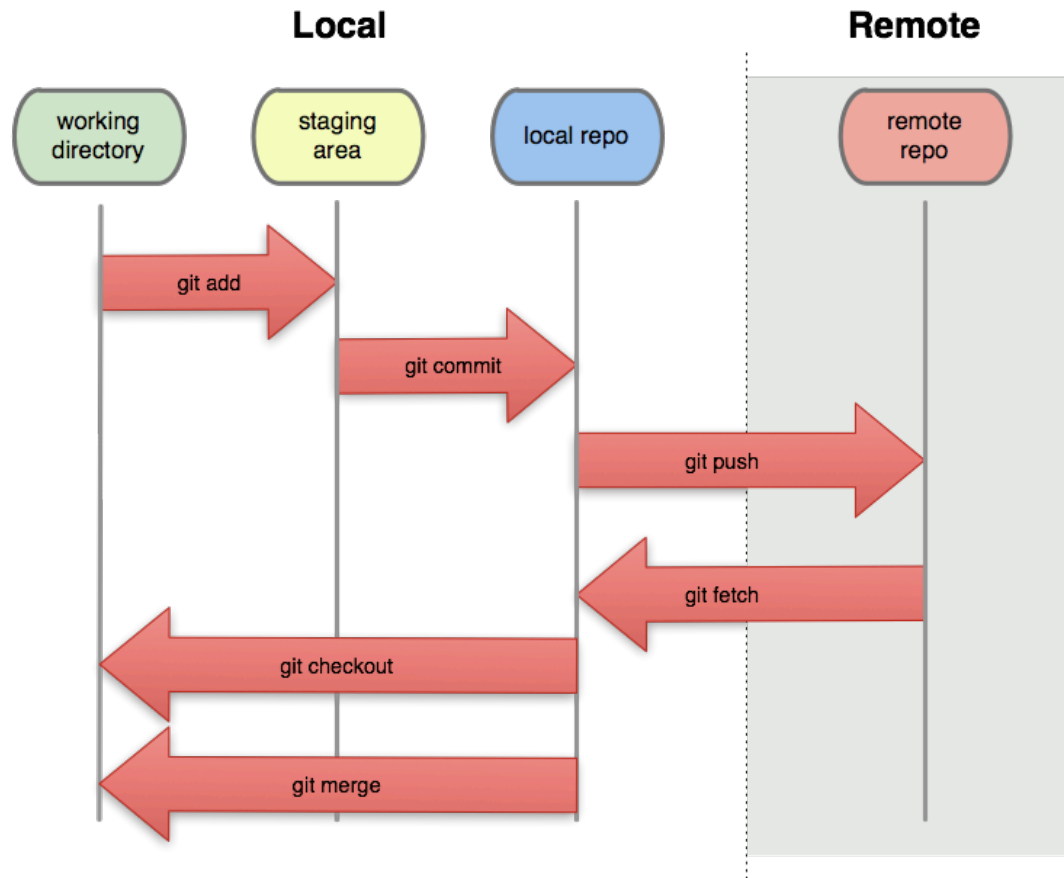
- Простая работа с ветками
- Развитые средства интеграции с другими VCS
- Продуманная система команд, позволяющая удобно встраивать git команды в скрипты
- Репозитории git могут распространяться и обновляться общесистемными файловыми утилитами, такими как rsync

Недостатки:

- Отсутствие переносимой на другие операционные системы поддержки путей в кодировке Unicode
- Команды ориентированы на наборы изменений, а не на файлы.
- Использование для идентификации ревизий хешей SHA1
- Проблемы с производительностью



# GIT



# Работа с GIT

## 0. Настройка Git

Задать конфигурации

```
$ git config --global --list
```

Изменить имя и email:

```
$ git config --global user.name "Super.User"
```

```
$ git config --global user.email "suser@supermail.com"
```

Определить протокол передачи данных:

- FILE — мы имеем прямой доступ к файлам репозитория.
- SSH — мы имеем доступ к файлам на сервере через ssh.
- HTTP(S) — используем http в качестве приёма/передачи.

# Работа с GIT

1. Клонировать или создать новый репозиторий или обновить текущий

```
$ git init
```

или

```
$ git clone git@github.com:user/repo.git
```

или

```
$ git pull
```

2. Сделать изменения

```
$ git status
```

```
# On branch master
```

```
# Untracked files:
```

```
# (use "git add <file>..." to include in what will be committed)
```

```
#
```

```
# README
```

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

# Работа с GIT

3. Сделать изменения и добавить их

```
$ git add README
```

```
$ git status
```

```
# On branch master
```

```
# Changes to be committed:
```

```
# (use "git reset HEAD <file>..." to unstage)
```

```
#
```

```
# new file: README
```

```
#
```

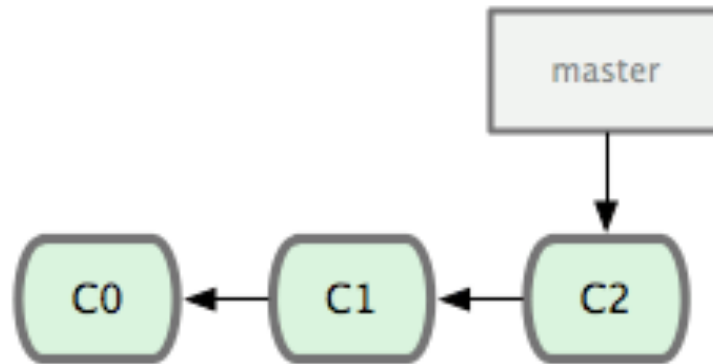
3. Commit changes:

```
$ git commit -m "Add README file"
```

4. Pull changes:

```
$ git push
```

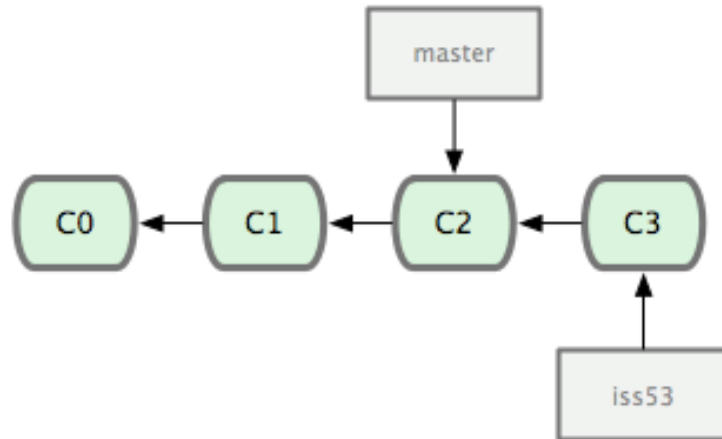
# Merge and GIT



```
$ git checkout -b iss53
```

Switched to a new branch "iss53"

# Merge and GIT



**\$ git checkout master**

Switched to branch "master"

**\$ git checkout -b hotfix**

Switched to a new branch "hotfix"

# Merge and GIT

```
$ git commit -m "Fix"
```

Switched to branch "master"

```
$ git checkout master
```

```
$ git merge hotfix
```

Updating f42c576..3a0874c

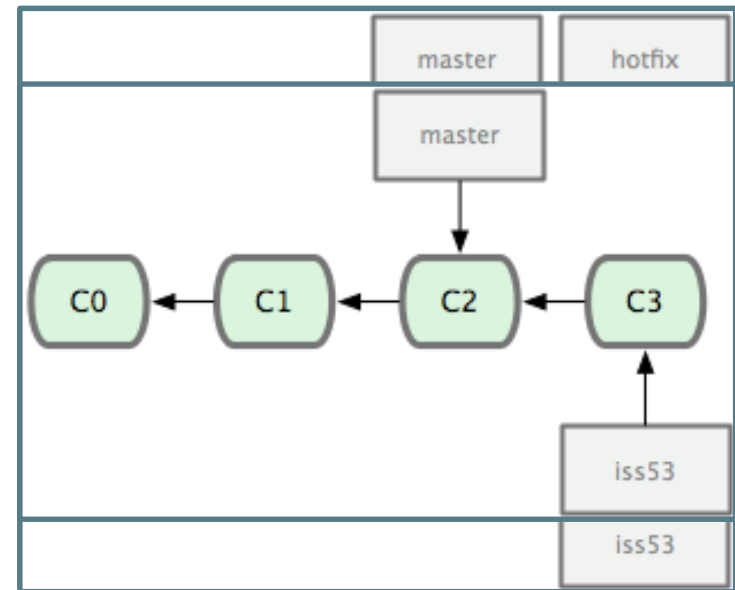
Fast forward

README | 1 -

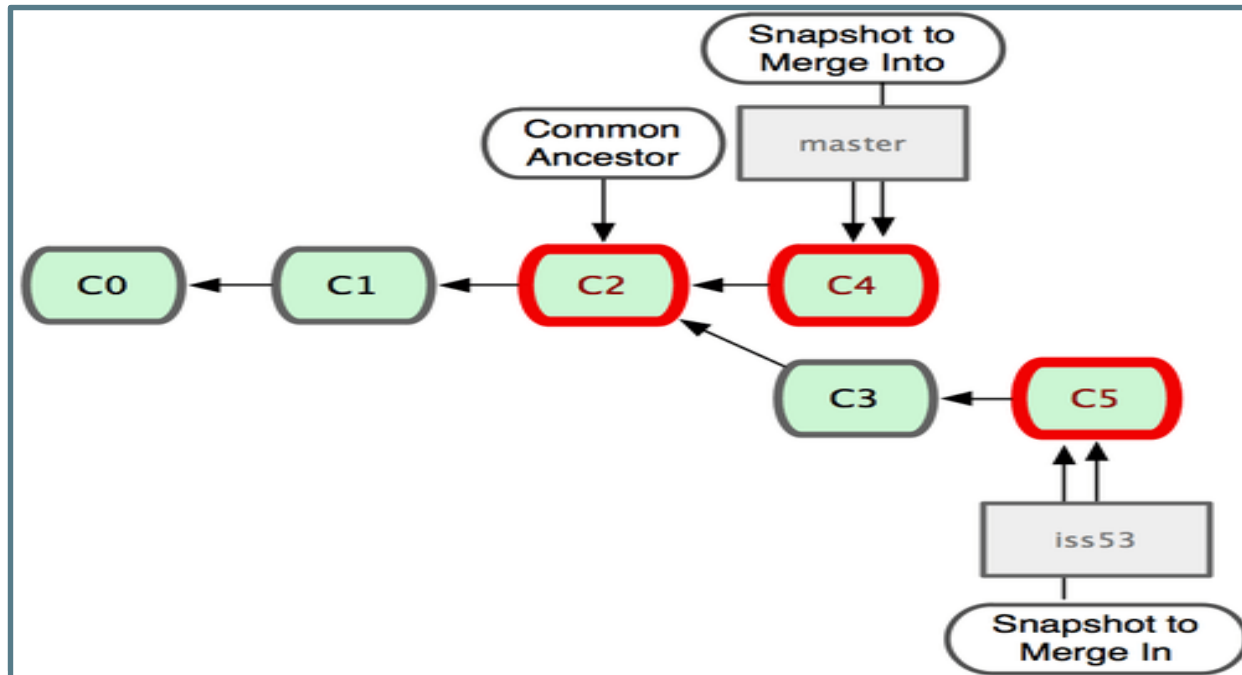
1 files changed, 0 insertions(+), 1  
deletions(-)

```
$ git branch -d hotfix
```

Deleted branch hotfix (3a0874c).



# Merge and GIT



```
$ git checkout master
```

```
$ git merge iss53
```

```
Merge made by recursive. README | 1 +
```

```
1 files changed, 1 insertions(+), 0 deletions(-)
```



# Разрешение конфликтов

```
$ git merge iss53
```

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

```
$ git status
```

```
index.html: needs merge
```

```
# On branch master
```

```
# Changes not staged for commit:
```

```
# (use "git add <file>..." to update what will be committed)
```

```
# (use "git checkout -- <file>..." to discard changes in working  
directory)
```

```
#
```

```
# unmerged: index.html
```

```
#
```

# Разрешение конфликтов

Не разрешенный конфликт в файле будет иметь вид:

```
<<<<<< 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
```

В верхней части блока (всё что выше =====) это версия из HEAD

После того, как вы разобрались со всеми конфликтами выполните:

```
$ git add
```

для каждого конфликтного файла. Индексирование будет означать для Git'a, что все конфликты в файле теперь разрешены.

Если вы хотите использовать графические инструменты для разрешения конфликтов, можете выполнить команду

```
$ git mergetool
```

# Разрешение конфликтов

```
$ git status
```

```
# On branch master
```

```
# Changes to be committed:
```

```
# (use "git reset HEAD <file>..." to unstage)
```

```
#
```

```
# modified: index.html
```

```
#
```

```
$ git commit
```

```
Сообщение по умолчанию:
```

```
Merge branch 'iss53'
```

```
Conflicts:
```

```
    index.html
```

```
#
```

```
# It looks like you may be committing a MERGE.
```

```
# If this is not correct, please remove the file
```

```
# .git/MERGE_HEAD
```

```
# and try again.
```

```
#
```

# Бесплатные VCS серверы

Есть много сервисов, которые предоставляют открытые репозитории для совместной работы:

- [sourceforge.net](https://sourceforge.net) — SVN, Git, Mercurial, Bazaar, CVS репозитории.
- [code.google.com](https://code.google.com) — SVN, Git, Mercurial репозитории.
- [github.com](https://github.com) — Git репозитории.
- [bitbucket.org](https://bitbucket.org) — Git и Mercurial репозитории.
- ...

# Полезные ссылки

Книга про Git с картинками:

<http://git-scm.com/book/ru/>

Подробная работа с Git:

<http://habrahabr.ru/post/174467/>

Хорошая модель ветвления в Git:

<http://habrahabr.ru/post/106912/>

Настоятельно рекомендую пройти курсы:

<http://try.github.io/levels/1/challenges/1>

<https://www.codeschool.com/courses/git-real>

Спасибо за внимание