Основы работы с распределенной системой контроля версий Git

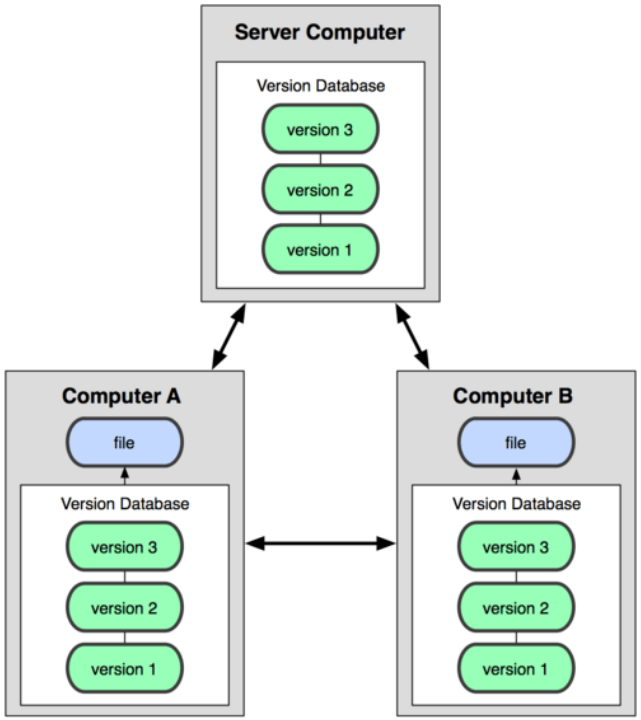
Илья Куприк

Лаборатория МФТИ-Интел

Назначение систем контроля версий

- Единое хранилище файлов проекта (кода / текста / картинок ...)
- Удобный доступ в хранилище для многих разработчиков
- Хранение истории изменений и возможность вернуться назад
- Поддержка специальных паттернов коллективной разработки (патчи, ветви проекта, слияние правок многих разработчиков ...)

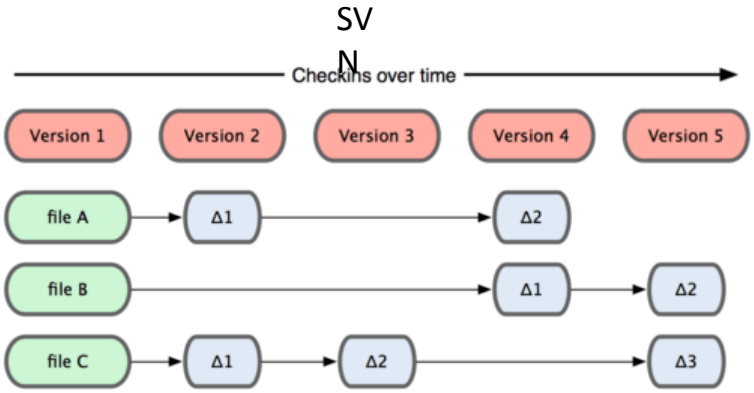
Git – распределенная система контроля версий

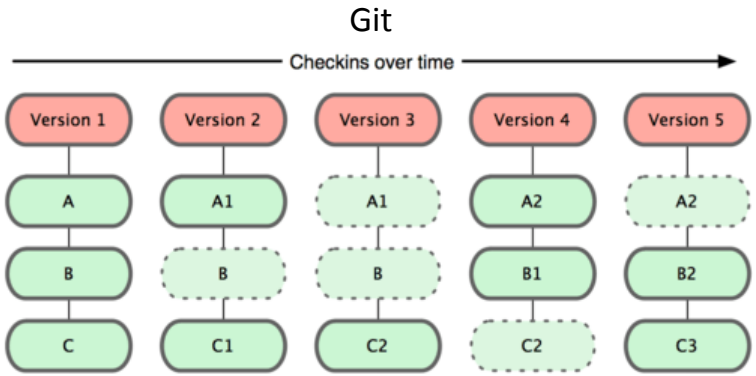


- Git широко используется для разработки больших проектов с большим количеством версий, ветвлений и различных сборок
- Ключевая особенность Git – способность тысячам разработчиков обмениваться патчами напрямую друг с другом, без необходимости проводить все изменения через единый сервер.

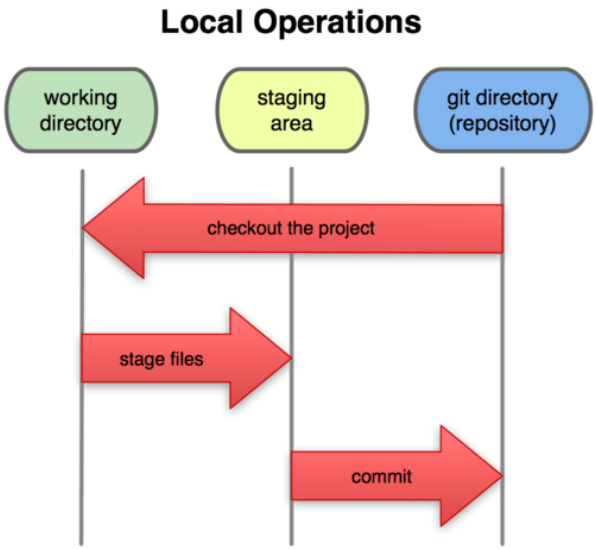
Организация хранилища Git

- Git – это мини файловая система, каждая версия в хранилище – это полноценный набор файлов (а не только изменений)
- Большинство операций (история, коммит, патч) – могут проводиться локально, т.к. Git хранит всю историю операций и копии файлов





Локальная среда разработки Git



- Рабочая директория – все файлы в директории проекта (включая версионированные и не версионированные)
- Файлы, подготовленные для добавления в репозиторий – файлы, которые разработчик пометил к добавлению (изменению) в репозиторий
- Файлы репозитория – текущий снимок последней зафиксированной копии проекта + вся история + системные файлы

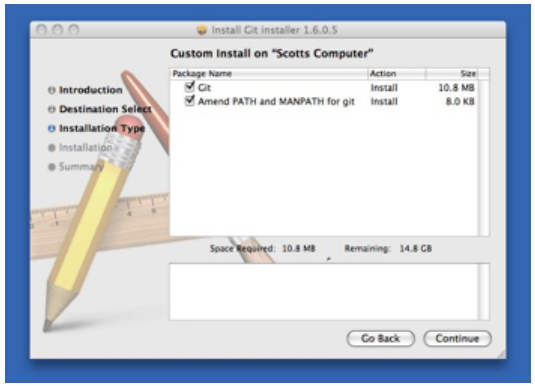
1. Изменяем файлы в рабочей директории (modify)
2. Помечаем что необходимо внести в репозиторий (stage)
3. Добавляем в репозиторий (commit)

Установка Git

- Debian-like: $ sudo apt-get install git
- OpenSuse: $ sudo zypper install git

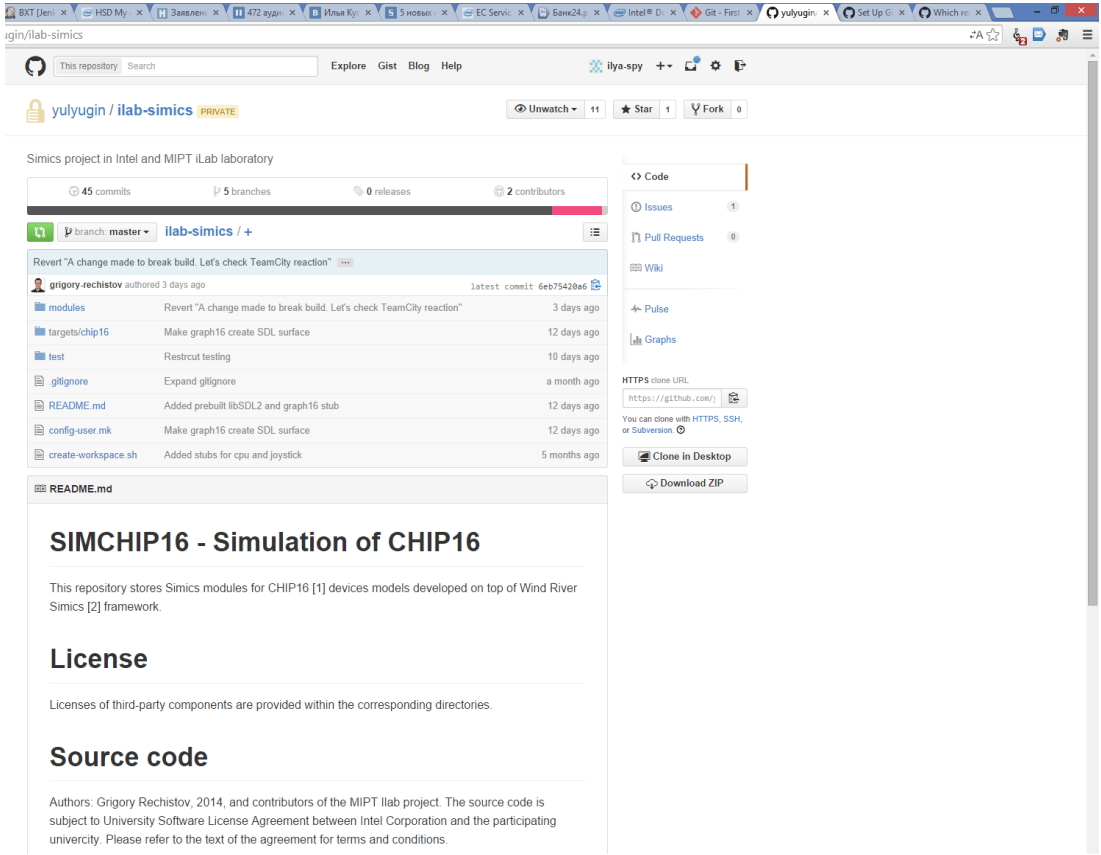
- Mac: http://sourceforge.net/projects/git-osx-installer/

- Windows: http://msysgit.github.io



GitHub – Git сервер с веб интерфейсом

- ivkuprik@IVKUPRIK-MOBL ~/Desktop
- $ **git clone https://github.com/yulyugin/ilab-simics.git**
- Cloning into 'ilab-simics'...
- Username for 'https://github.com': ilya-spy@yandex.ru
- Password for 'https://ilya-spy@yandex.ru@github.com':
- remote: Counting objects: 653, done.
- remote: Compressing objects: 100% (280/280), done.
- remote: Total 653 (delta 336), reused 637 (delta 326)
- Receiving objects: 100% (653/653), 2.58 MiB | 105.00 KiB/s, done.
- Resolving deltas: 100% (336/336), done.
- Checking connectivity... done.



Основные операции с репозиториями

Git Init in current folder:

```
$ git init
```

Git add files to stage area (mark as tracked)

```
$ git add *.c
$ git add README
```

Git commit files into new revision of local repository

```
$ git commit -m 'initial project version'
```

Git clone existing repository in current folder

```
$ git clone git://github.com/schacon/grit.git
```

$ vim benchmarks.rb

$ **git status**

On branch master

Changes to be committed:

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

new file: README

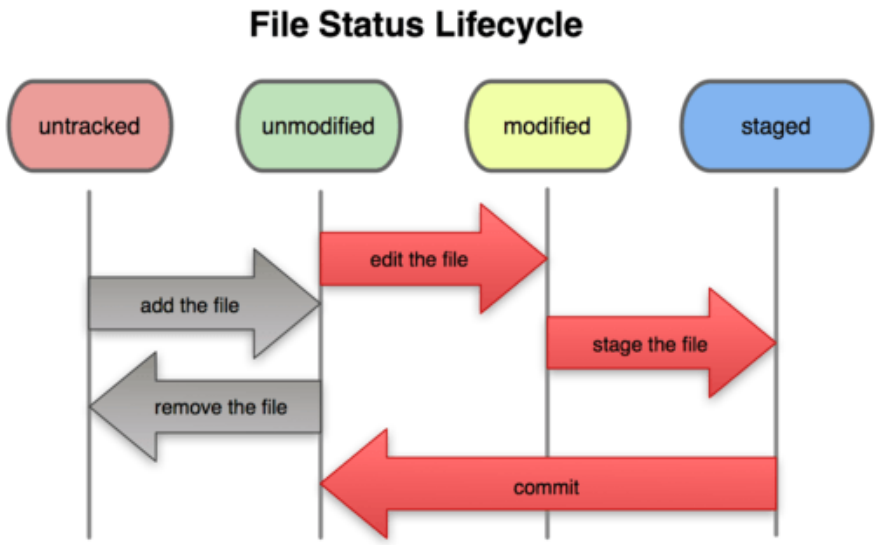
modified: benchmarks.rb

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)

modified: benchmarks.rb

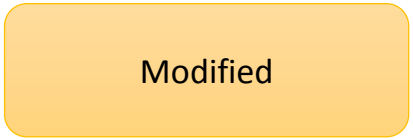


Просмотр файлов в репозитории Git

.gitignore

```
# a comment - this is ignored
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a # files above
!lib.a
# only ignore the root TODO file, 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 .txt files in the doc/ directory
doc/**/*.txt
```

$ git diff показывает файлы



$ git diff --satged (--cached) показывает файлы

Фиксирование изменений и удаление файлов

- `$ git commit -m "Story 182: Fix benchmarks for speed"`

- Пропуск стадии staging:

$ git status

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)

modified: benchmarks.rb

no changes added to commit (use "git add" and/or "git commit -a")

$ git commit -a -m 'added new benchmarks'

Удаление файла из репозитория и с жесткого диска

```
$ git rm grit.gemspec
$ git commit –m “Removing file”
```

Удаление файла только из репозитория

```
$ git rm --cached readme.txt

$ git commit –m “Removing file”
```


iLab

Работа с изменениями

- **$ git log**

commit 6eb75420a67d76afe7e753d098a89e2a2dd21d83

Author: Grigory Rechistov <grigory.rechistov@phystech.edu>

Date: Sun Sep 14 03:35:50 2014 +0400

Revert "A change made to break build. Let's check TeamCity reaction"

This reverts commit 221d0e7f59b06c26f71b450a0d5bbae49e335b96.

commit 221d0e7f59b06c26f71b450a0d5bbae49e335b96

Author: Grigory Rechistov <grigory.rechistov@phystech.edu>

Date: Sun Sep 14 03:34:41 2014 +0400

A change made to break build. Let's check TeamCity reaction

commit 82243d4540e185d2e034f3c72431280b2312bc66

Author: Grigory Rechistov <grigory.rechistov@phystech.edu>

Date: Sun Sep 14 03:28:50 2014 +0400

Test change to trigger successful build

Unstage changes:

$ git reset HEAD benchmarks.rb

$ git status

On branch master

Changes to be committed:

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

modified: README.txt

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)

modified: benchmarks.rb

Discard local changes

$ git checkout -- benchmarks.rb

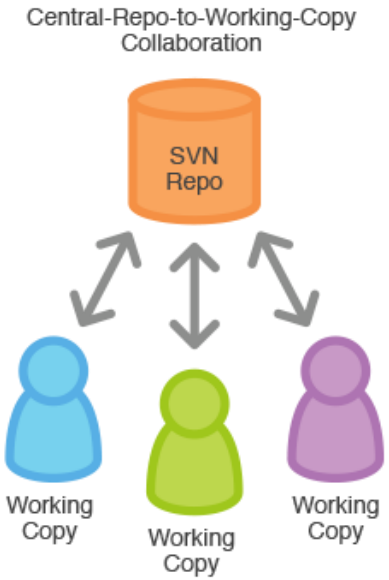
$ git status

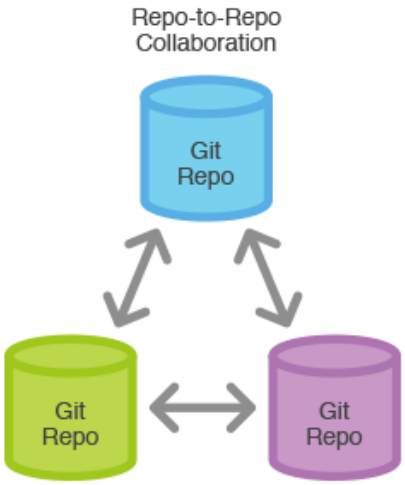
On branch master

Changes to be committed:

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

Работа с удаленными репозиториями





$ git remote -v

Backdoor git://github.com/bakkdoor/grit.git

cho45 git://github.com/cho45/grit.git

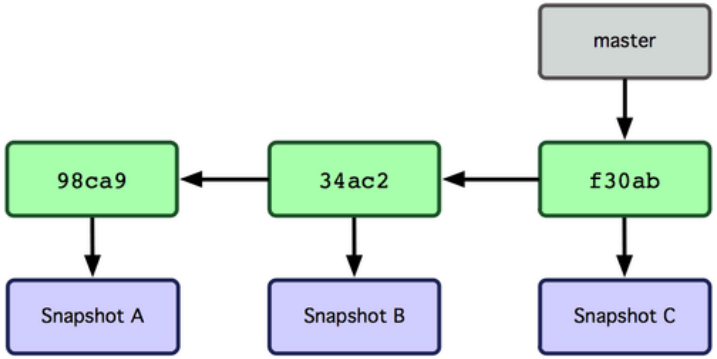
defunkt git://github.com/defunkt/grit.git

koke git://github.com/koke/grit.git

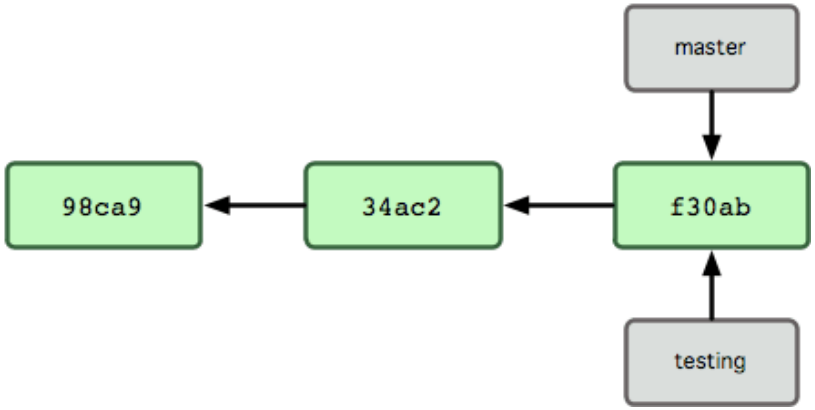
origin *git@github.com:mojombo/grit.git*

```
$ git fetch [remote-name]
$ git push [remote-name][branch-name]
$ git push origin master
```

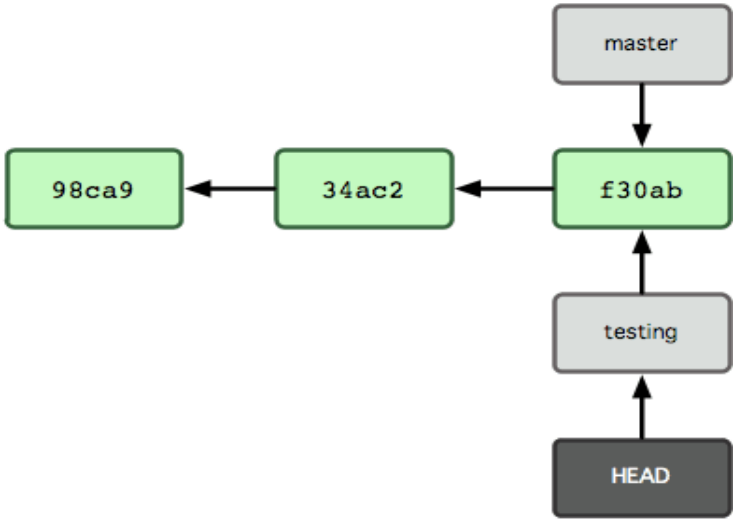
Ветвления в репозиториях Git

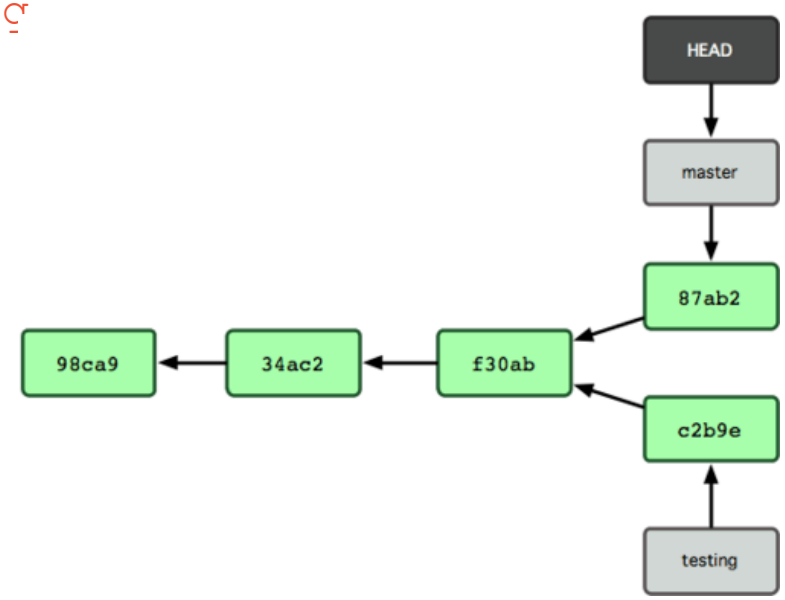



```
$ git branch testing
```

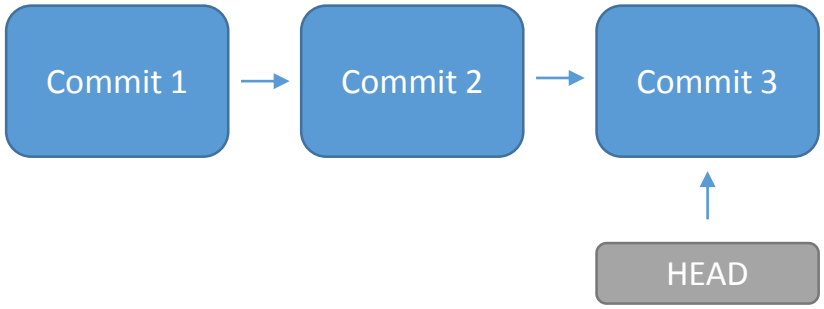



```
$ git checkout testing
```

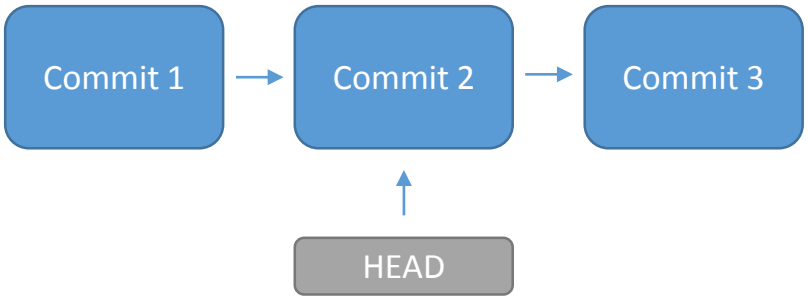




Работа с ошибочными коммитами

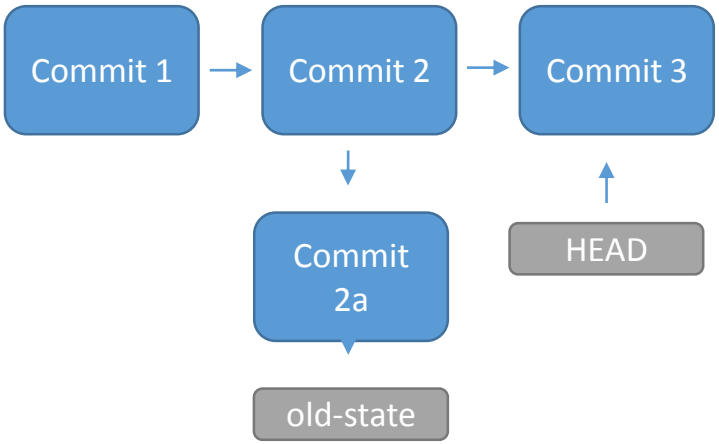



```
# This will detach your HEAD,
that is, leave you with no
branch checked out:

git checkout 2
```

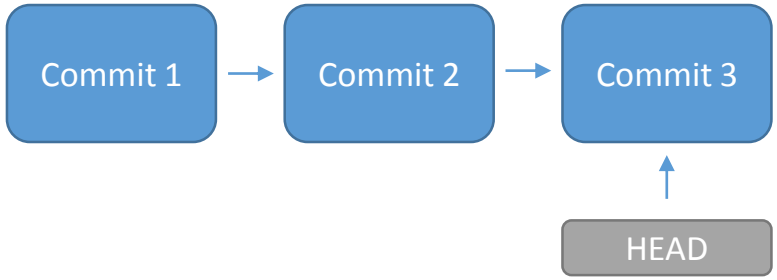


Or if you want to make commits while you're there, go ahead and make a new branch while you're at it:

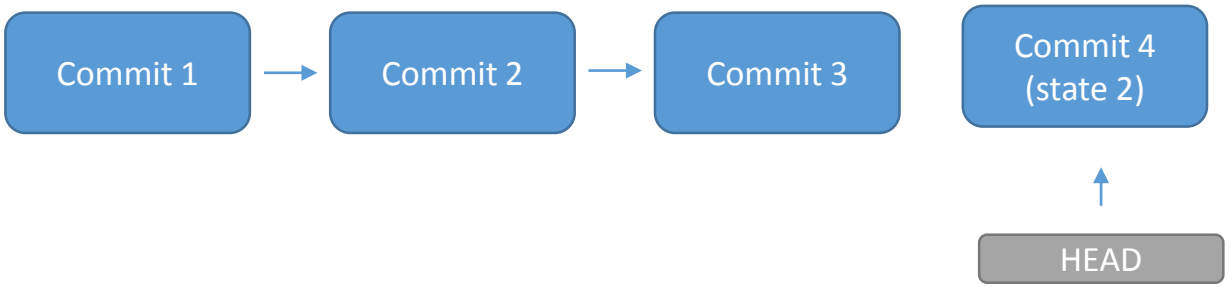
```
git checkout -b old-state 2
touch a.txt

git commit –m “Starting new
branch from old state”
```



Откат к старому состоянию и перезапись HEAD




```
# This will create three separate revert commits:
git revert 3

# It also takes ranges. This will revert the last two
commits:
git revert HEAD~2..HEAD
```



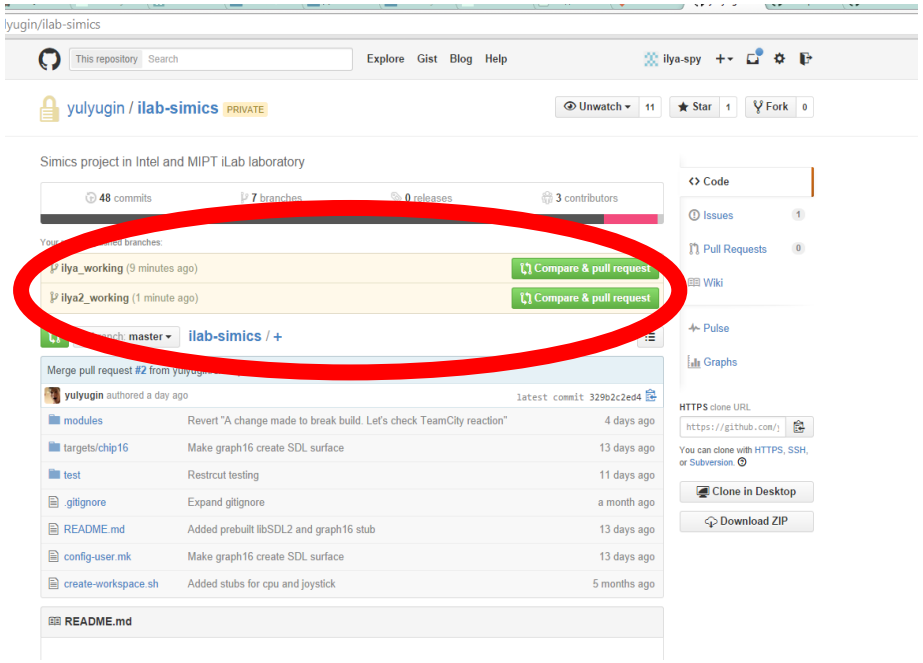
Цикл разработки в Git / GitHub

- Клонировать текущее состояние репозитория проекта из GitHub
- Создать локальные ветки (branches) для разработки отдельных заданий / функций
- Накапливать изменения путем коммитов в бранчи (октатывать в случае необходимости)
- Когда задание готово к отправке в основной репозиторий – убедиться что все тесты проходят в бранче и:
  - Опция 1 (нам не подходит): произвести слияние бранча с master и закоммитить в master
  - Опция 2 (наш случай): закоммитить свой бранч непосредственно в GitHub репозиторий, зайти на сайт github.com и через веб-интерфейс послать запрос на слияние закоммиченного вами бранча с master

Запросы на слияние изменений (Pull requests)

- Для контроля за качеством кода / упорядочения процесса разработки часто вводится запрет на непосредственно коммиты в мастер (основную ветвь) проекта
- Назначаются ответственные люди, которые проверяют каждый патч и производят его **слияние** с мастером
- В этом случае необходимо:
  - Закоммитить свои изменения в бранч на удаленном сервере

    $ git push origin your_working_branch_name
  - Послать запрос ответственному лицу на слияние ваших правок:



INTEL CORPORATION
UNIVERSITY SOFTWARE LICENSE AGREEMENT
IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING.

- This Agreement is granted for a specific department, a declared course, or a specific computer lab as specified by Intel.
- The Software may only be used for research projects that are non-funded, or funded by a non-profit organization.
- The Licensed Materials contain confidential and/or proprietary information of Intel and, together with the terms of this Agreement, are subject to the Non-Disclosure Agreement
- Recipient shall not incorporate any Open Source Software <…> into the Licensed Software in any manner or take any steps which in any manner would cause the Licensed Software to be subject to any license obligations associated with Open Source Software.

Правило №1: если вы не уверены — спросите ментора!

iLab

Литература

- Scott Chason. ProGit (русская версия)
- Интерактивный тур по Git Git How To

Спасибо за внимание!

Слайды и материалы курса доступны по адресу http://is.gd/YAc1ll