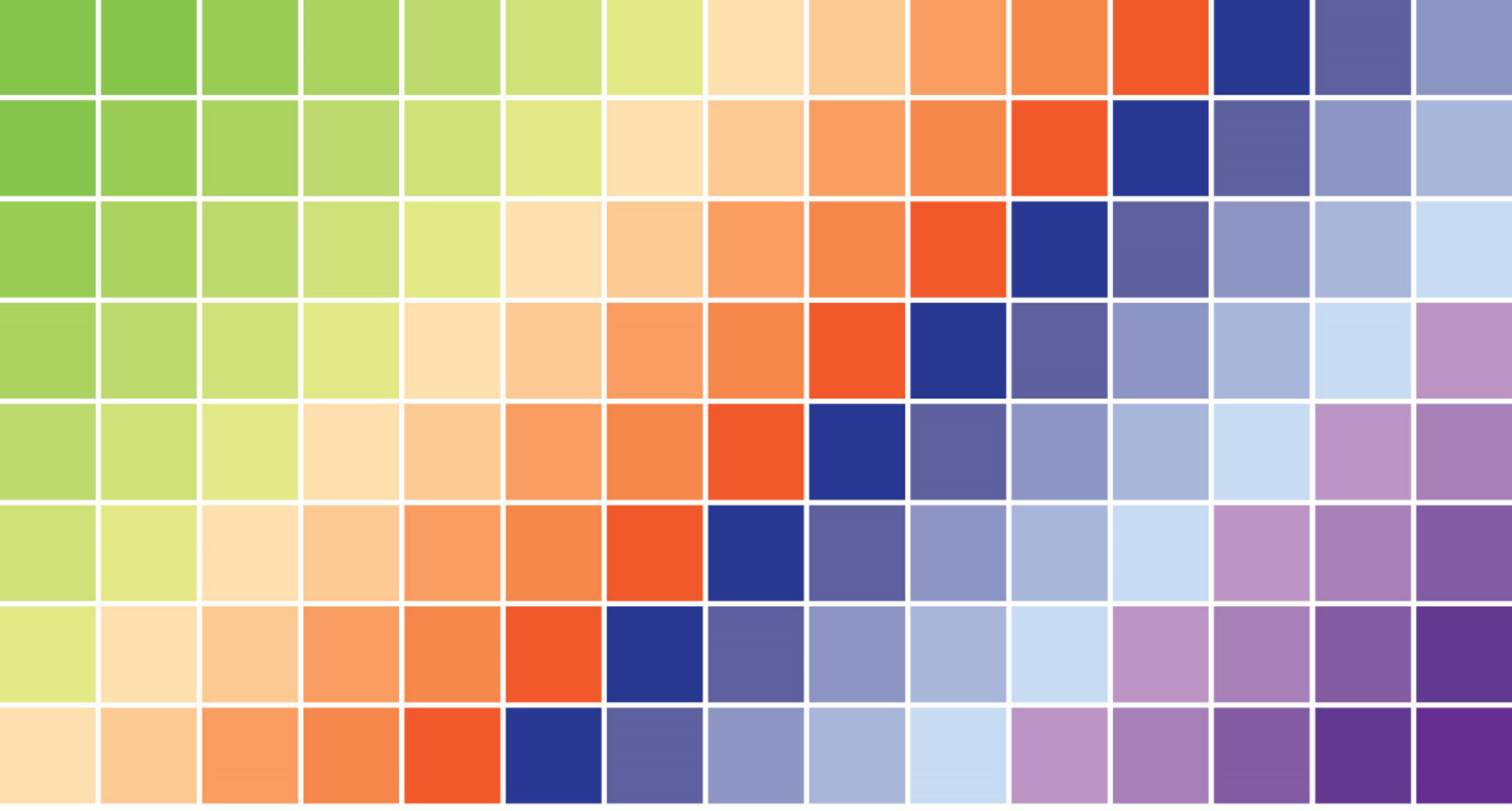


School of Computer Science

СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ОПРЕДЕЛЕНИЕ И КЛАССИФИКАЦИЯ

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов

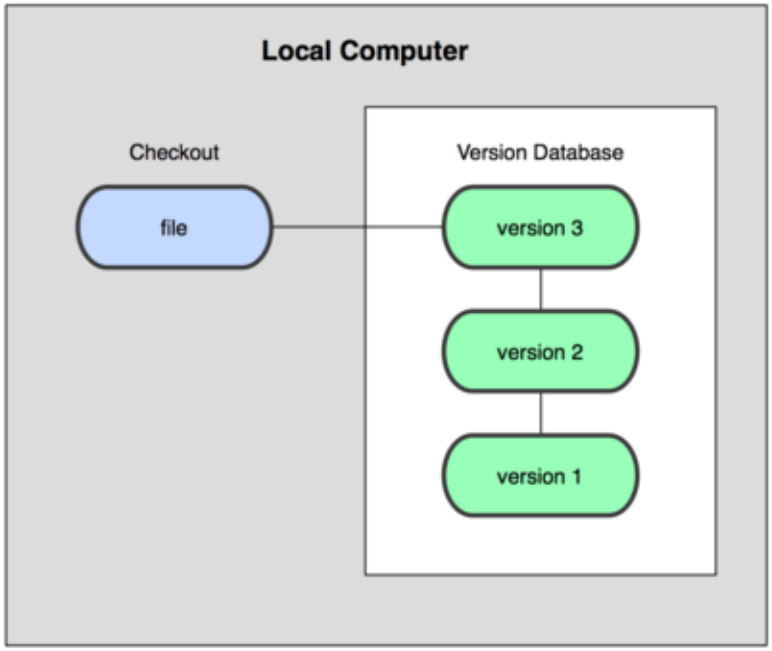
ЛОКАЛЬНЫЕ СКВ

Суть: локальная база данных, в которой хранится список изменений исходного файла

Особенности:

- Работа только с одним файлом;
- Невозможность одновременной работы нескольких пользователей с системой;
- Риск потери данных.

Представители: RCS (Revision Control System)



ЦЕНТРАЛИЗОВАННЫЕ СКВ

Суть: центральный сервер, на котором хранятся все файлы под версионным контролем; клиенты получают копии файлов

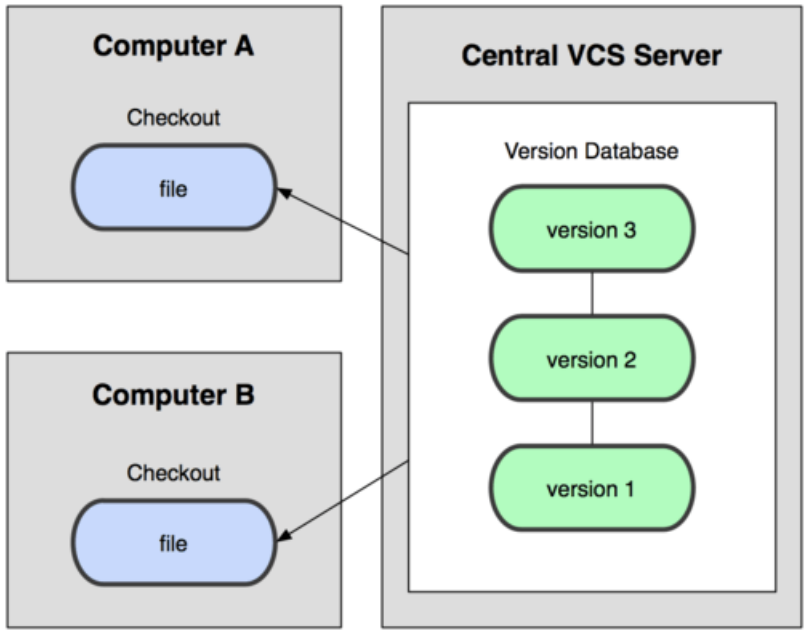
Плюсы:

- Под контролем находятся все файлы;
- Возможность работы нескольких пользователей;
- Удобство администрирования.

Минусы:

- Риск потери данных.

Представители: CVS, SVN



Компас Плюс использует для работы SVN

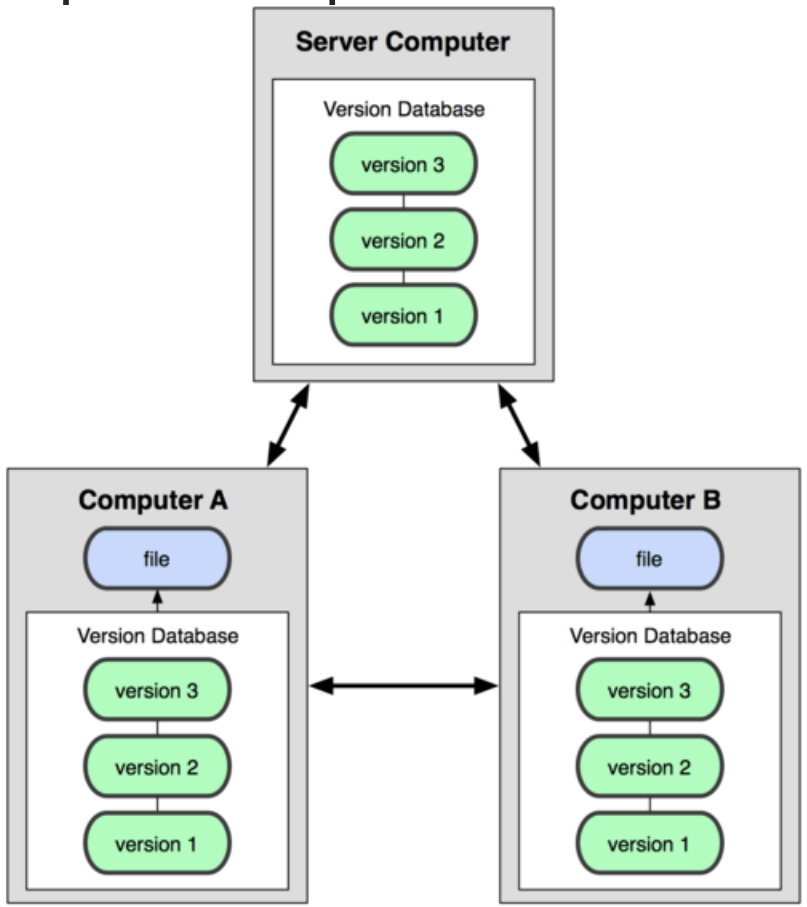
РАСПРЕДЕЛЕННЫЕ СКВ

Суть: клиенты забирают весь репозиторий, который в любой момент может быть скопирован обратно на сервер

Особенности:

- Отсутствие четко выделенного центрального хранилища версий - репозитория
- Возможность работы с несколькими удалёнными репозиториями

Представители: Git, Mercurial



ЧТО ТАКОЕ GIT

Git – распределенная система контроля версий

Разработан: Линусом Торвальдсом в 2005 году

Проекты: ядро Linux, Swift, Android, jQuery, PHP, Qt, ряд дистрибутивов Linux

Сервисы для работы: GitHub, GitLab, TortoiseGit



РЕПОЗИТОРИИ

Репозиторий – место, где хранятся и поддерживаются какие-либо данные

- **Локальный** – место на вашем компьютере
- **Удаленный** – хранилище в сети (например, GitHub)

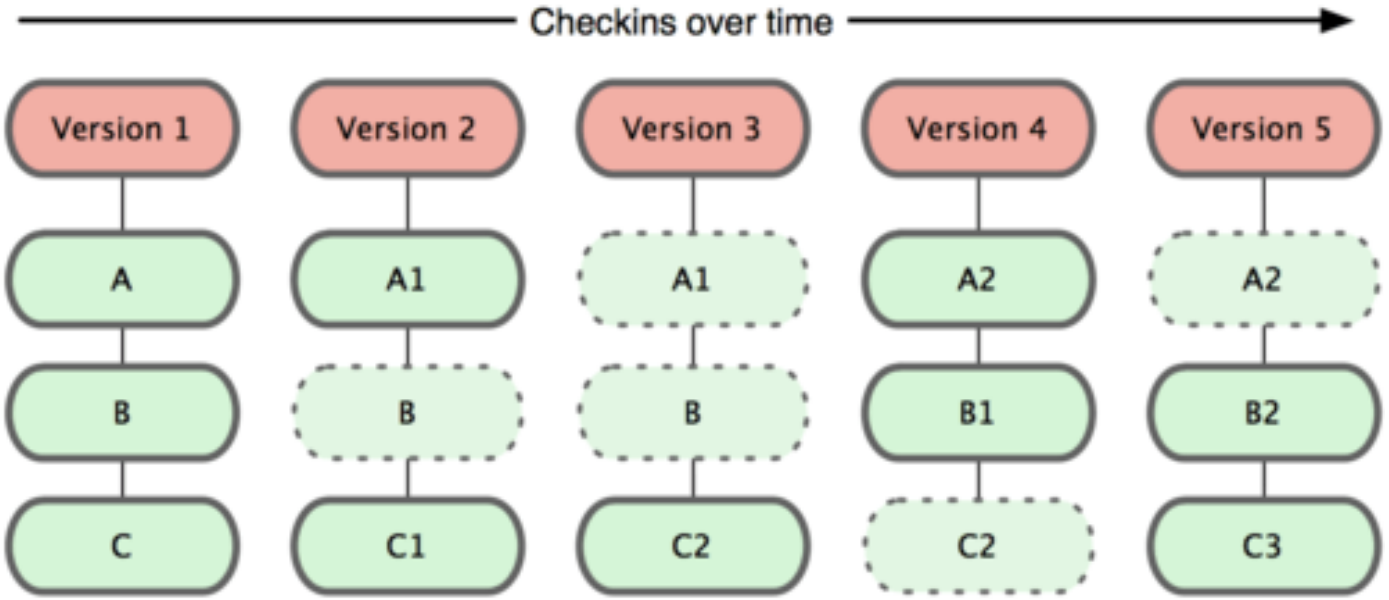
Удаленный репозиторий принято называть origin

ХРАНЕНИЕ ДАННЫХ

Слепок - состояние всех файлов рабочего каталога на текущий момент времени

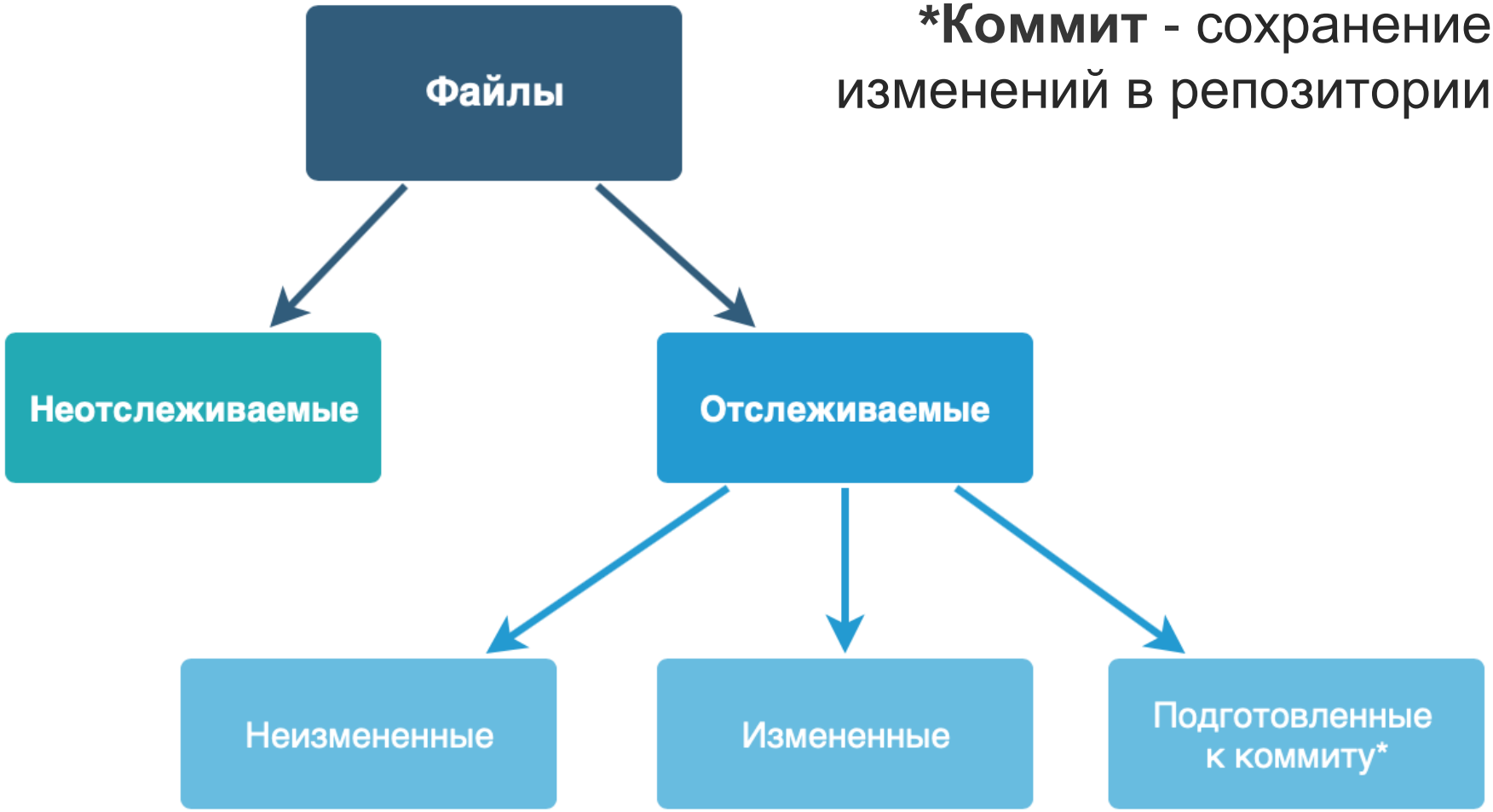
Git: хранит файлы в виде слепков

Другие СКВ: хранят список изменений для файлов



СОСТОЯНИЯ ФАЙЛОВ

***Коммит** - сохранение изменений в репозитории



НАЧАЛО РАБОТЫ

- Установить git:

  https://git-scm.com/downloads
- Перейти в командную строку
- Указать данные пользователя:

```
$ git config --global user.name ”UserName”
```

```
$ git config --global user.email myname@example.com
```

- Перейти в каталог проекта и создать в нем git-репозиторий:

```
$ git init
```

ОСНОВНЫЕ КОМАНДЫ

- Текущее состояние репозитория:

```
$ git status
```

- Добавить файл под версионный контроль/к коммиту:

```
$ git add filename
```

- Коммит:

```
$ git commit –m ”commit message”
```

- История изменений:

```
$ git log
```

- Удаление файла:

```
$ git rm
```

- Commit message должен быть осмысленным и четко передавать содержание коммита
- Одна доработка – один коммит

ИГНОРИРОВАНИЕ ФАЙЛОВ

- Что игнорировать?

  Автоматически генерируемые и временные файлы (логи, кэш, результаты сборок и т.д.)

- Как?

  Добавить в проект файл **.gitignore** с перечислением шаблонов игнорируемых файлов:

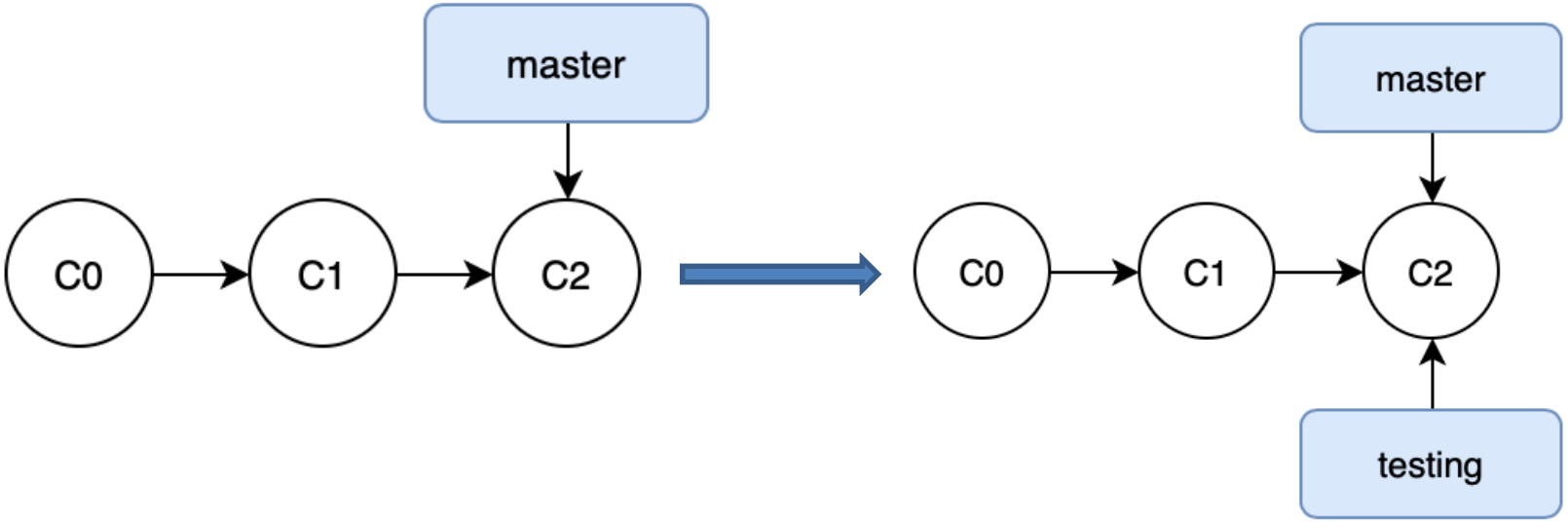
```
$ git add .gitignore
```

Сборник готовых .gitignore файлов:
https://gist.github.com/octocat/9257657

Ветка (branch) – это подвижный указатель на один из коммитов. Ветка по умолчанию называется *master.*

Создание новой ветки:

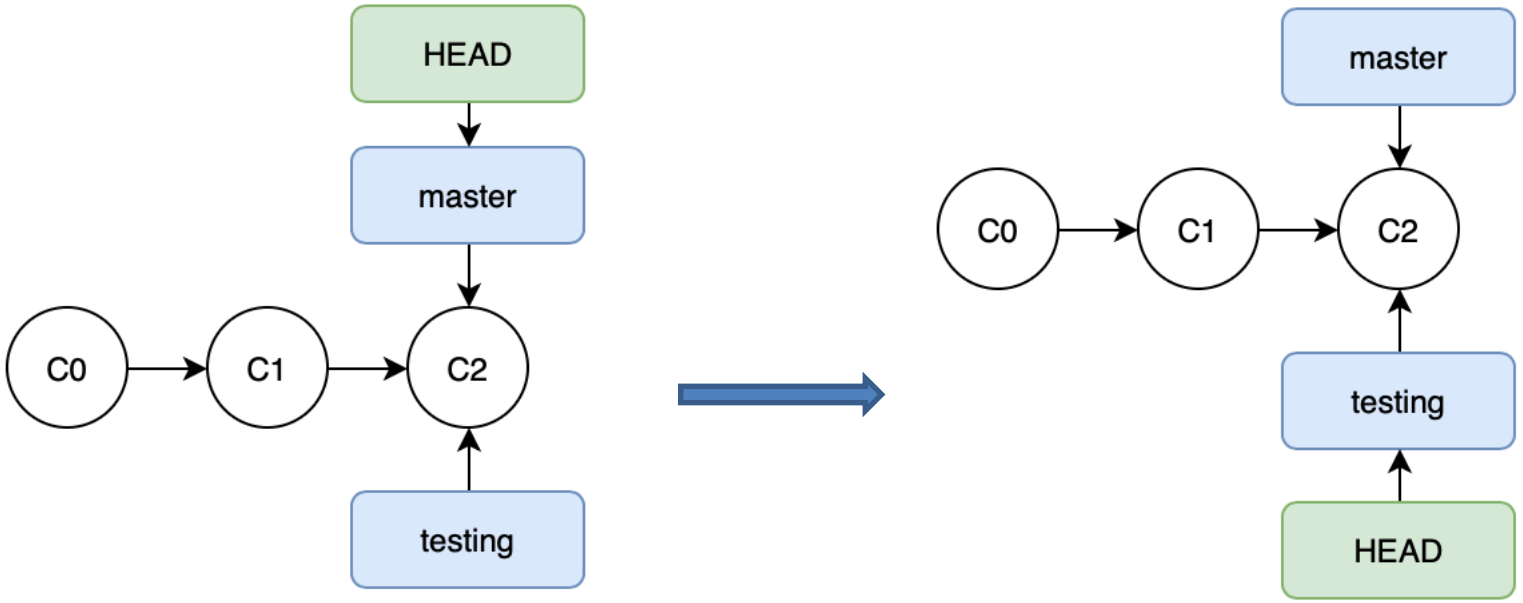
```
$ git branch testing
```



HEAD – это указатель на текущую ветку

Перейти на другую ветку:

```
$ git checkout testing
```



СЛИЯНИЕ

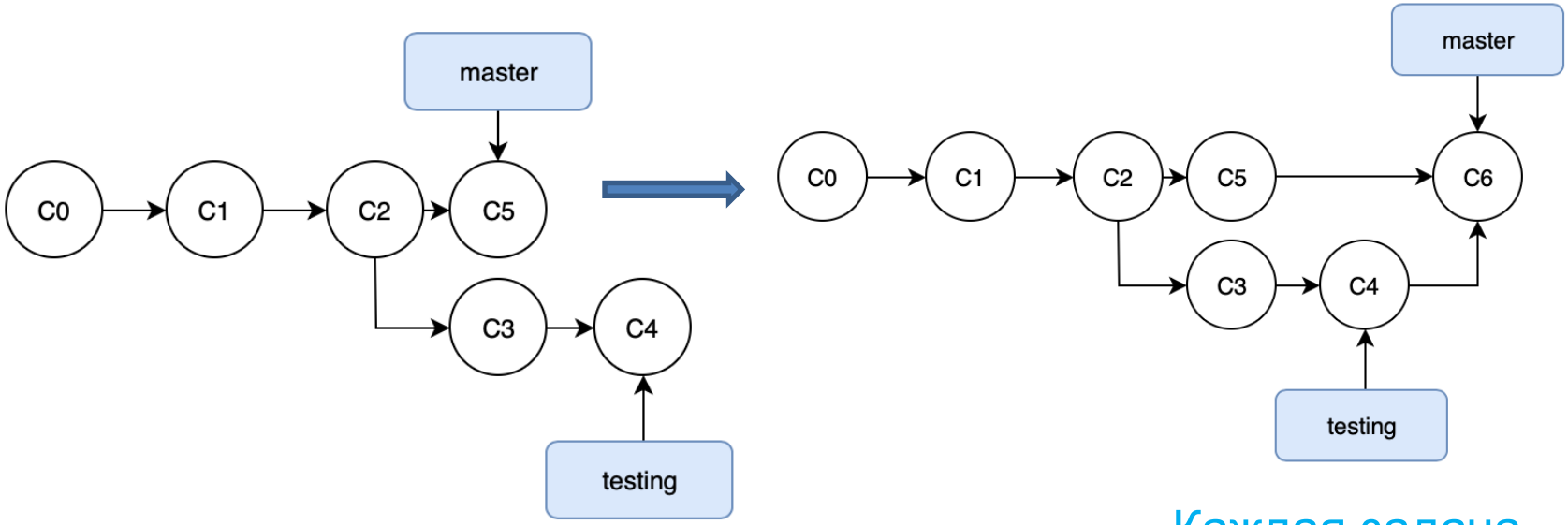
Возврат на основную ветку:

```
$ git checkout master
```

Слияние веток:

```
$ git merge testing
```

При слиянии возможны конфликты!



Удаление ветки:

```
$ git branch -d testing
```

Каждая задача решается в своей ветке

РАБОТА С УДАЛЕННЫМ РЕПОЗИТОРИЕМ

- Добавить удаленный репозиторий:

```
$ git remote add url
```

- Клонирование удаленного репозитория:

```
$ git clone url
```

- Получение и слияние данных:

```
$ git pull [репозиторий][ветка]
```

- Только получение данных:

```
$ git fetch [репозиторий][ветка]
```

- Отправка данных:

```
$ git push [репозиторий][ветка]
```

```
$ git push origin master
```

GITHUB И GITLAB

GitHub и **GitLab** – сервисы для онлайн-хостинга проектов, использующих Git
Цель создания: содействовать взаимодействию разработчиков

Функции:

- Разработка open-source проектов;
- Контроль доступа;
- Багтрекинг;
- Управлением задачами;
- Документация к проекту;

Страничка на GitHub – лицо разработчика

ПОЛЕЗНЫЕ ССЫЛКИ

Книга “Pro Git” с переводом на русский:
https://git-scm.com/book/ru/v2

Интерактивный курс по использованию Git:
https://githowto.com

Шпаргалка по командам:
https://eax.me/git-commands/

Статья ”Git – советы новичкам”:
https://habr.com/en/company/playrix/blog/345732/

Визуальный клиент для работы с git (только для Windows):
https://tortoisegit.org

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science