# Технологии разработки программного обеспечения

2017 / 2018, 1 курс, 2 семестр

Пудов Сергей Григорьевич

# Структура курса

- Инструментарий для командной разработки ПО: системы контроля версий
  - CVS
  - SVN
  - ***Git***
- Обеспечение качества ПО – тестирование
  - ***Unit***-тесты
  - Regression-тесты
  - …
- Работа в команде над выпуском продукта
  - Курсовая работа: команда 2-3 чел

# Лекция 1

- Системы контроля версий: история создания
  - RCS
  - CVS
  - SVN
  - Git
- Git на локальной машине:
  - Git add
  - Git commit
  - Git status

# Локальные системы контроля версий

**RCS (Revision Control System, Система контроля ревизий)**

была разработана в начале 1980-х годов Вальтером Тичи (Walter F. Tichy). Система позволяет хранить версии только одного файла, таким образом управлять несколькими файлами приходится вручную.

*ci (от check-in, регистрировать):*

*$ ci file.txt*

*co (от check-out)*

*$ co file.txt*

Недостатки:

- Работа только с одним файлом, каждый файл должен контролироваться отдельно;
- Неудобный механизм одновременной работы нескольких пользователей с системой

# Централизованные системы контроля версий: CVS

***CVS (Concurrent Versions System, Система совместных версий)***

Дик Грун (Dick Grune) разработал CVS в середине 1980-х. Для хранения индивидуальных файлов CVS (также как и RCS) использует файлы в RCS формате, но позволяет управлять группами файлов расположенных в директориях. Также CVS использует клиент-сервер архитектуру в которой вся информация о версиях хранится на сервере.

*$ cvs checkout path-in-repository* *co = checkout*

*$ cvs commit -m "Some changes“* *ci = commit*

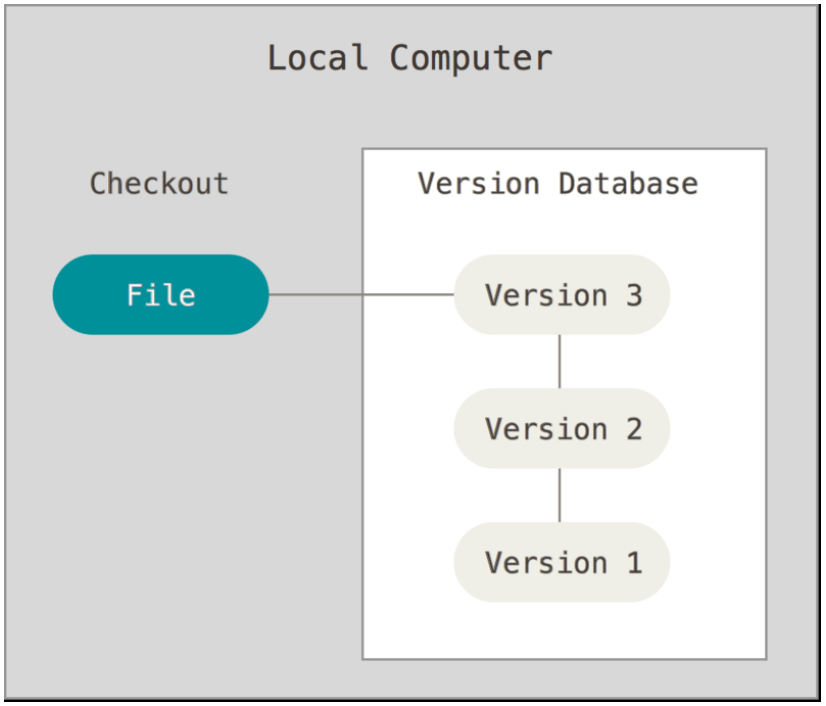
*$ cvs update*

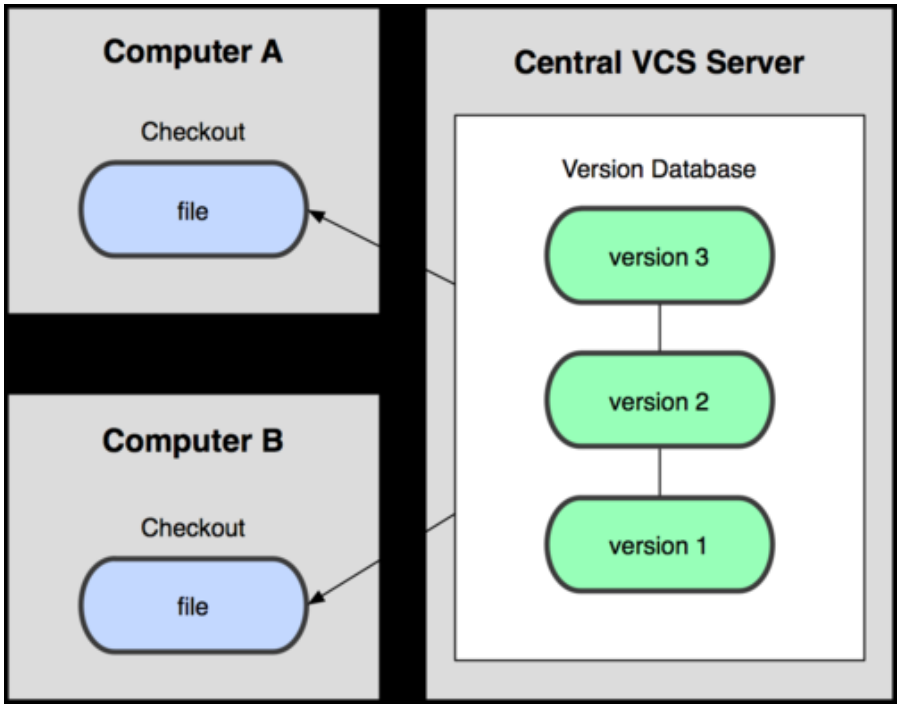
Недостатки:

Так как версии хранятся в файлах RCS нет возможности сохранять версии директорий.

Перемещение, или переименование файлов не подвержено контролю версий.

# Системы контроля версий:



# Централизованные системы контроля версий: SVN

***SVN (Subversion)*** *свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet*

Из наиболее значительных изменений по сравнению с CVS можно отметить:

- Атомарное внесение изменений (commit). В случае если обработка коммита была прервана не будет внесено никаких изменений.
- Переименование, копирование и перемещение файлов сохраняет всю историю изменений.
- Директории, символические ссылки и мета-данные подвержены контролю версий.
- Эффективное хранение изменений для бинарных файлов.

Команды SVN во многом похожи на CVS:

*$ svn checkout path-in-repository*

*$ svn commit -m "Some changes“*

*$ svn up*

# Распределенные системы контроля версий: git (2005)

В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных.

Основные требования к новой системе были следующими:

- Скорость
- Простота дизайна
- Поддержка нелинейной разработки (тысячи параллельных веток)
- Полная распределённость
- Возможность эффективной работы с такими большими проектами, как ядро Linux (как по скорости, так и по размеру данных)

# Git на локальной машине:

Книга: https://git-scm.com/book/ru/v2

Помощь:

- *$ git help <verb>*
- *$ git <verb> --help*
- *$ man git-<verb>*

Создание репозитория:

- *$ git init*

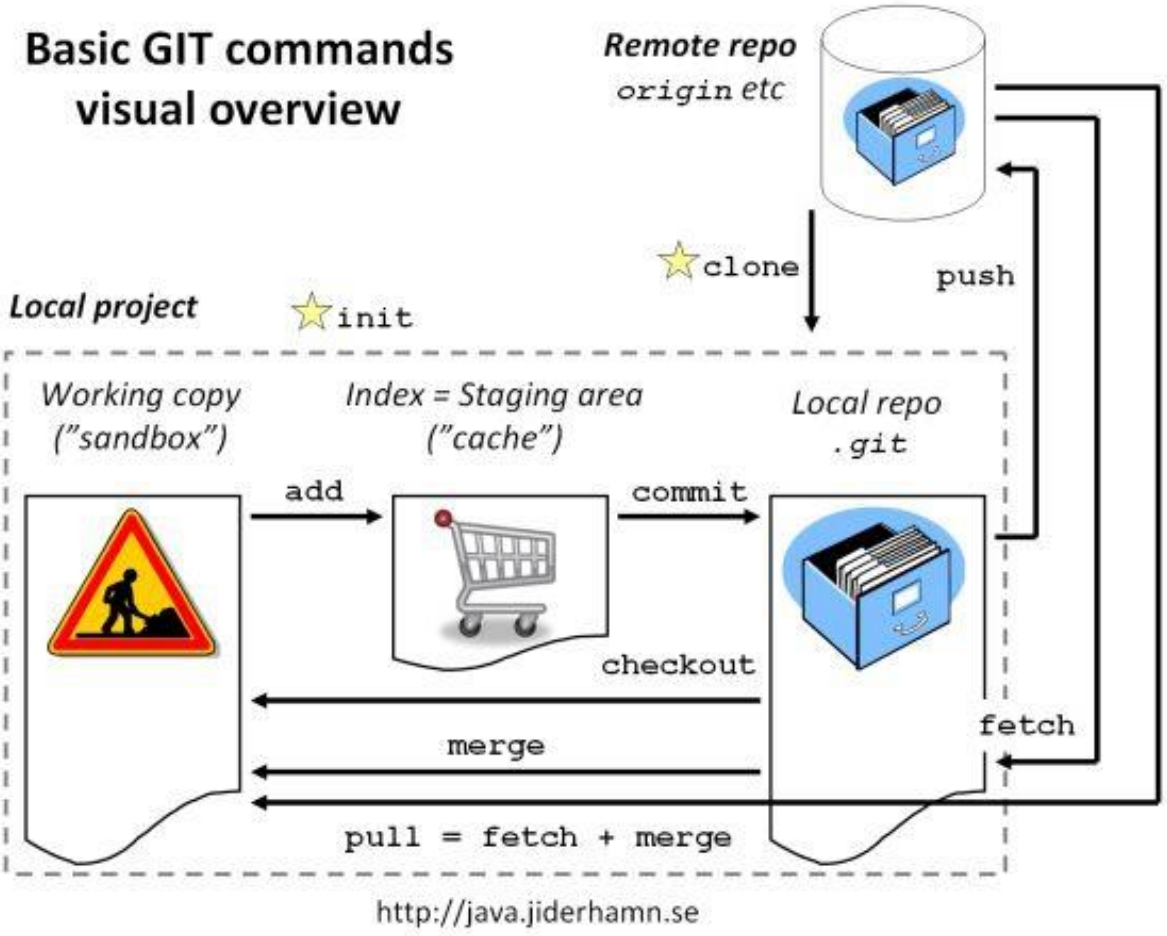
Эта команда создаёт в текущей директории новую поддиректорию с именем .git

Добавление файлов под версионный контроль:

- *$ git add <filename>*

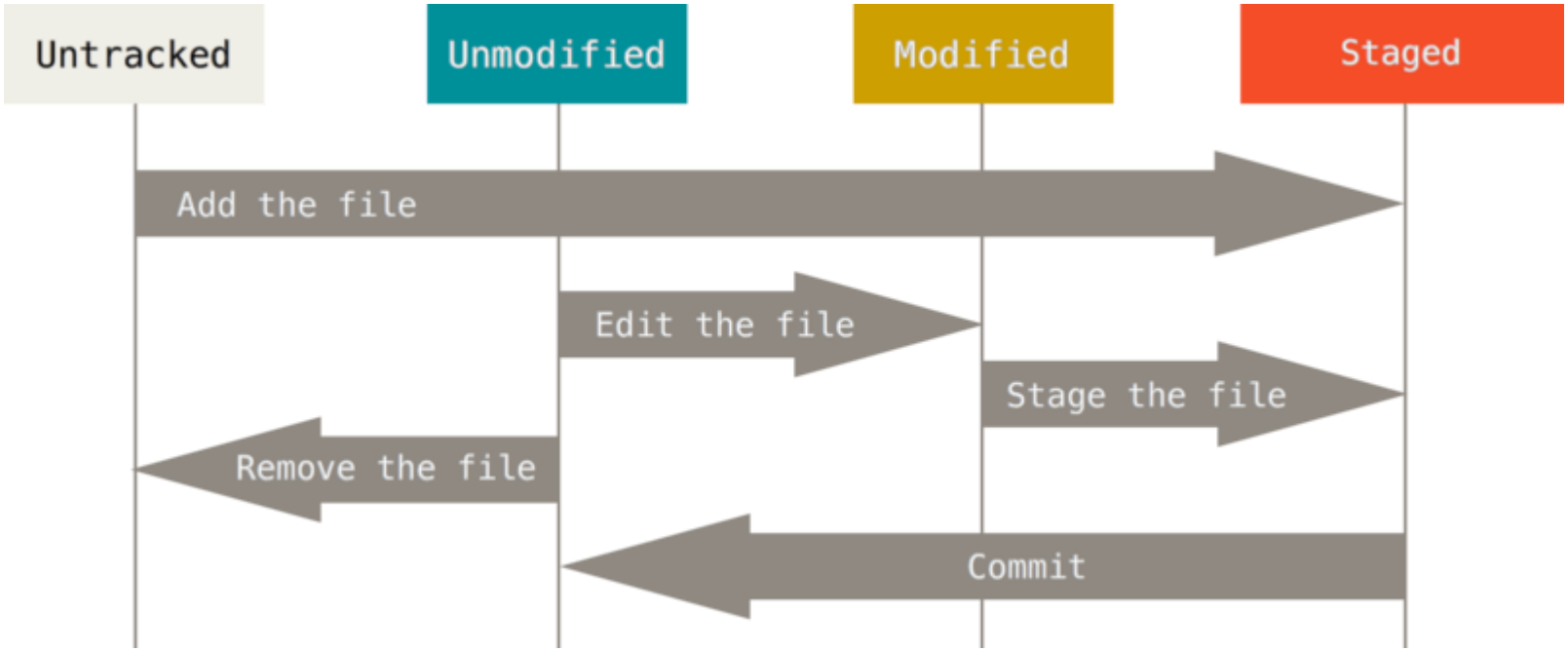
- *$ git commit -m 'initial project version'*

# Git на локальной машине:




http://www.about-dev.com/version-control/44-git-intro

# Git на локальной машине:



# Git на локальной машине:

Определение состояния файлов:

- *$ git status*

  *On branch master*

  *nothing to commit, working directory clean*

После создания файла в каталоге:

- *$ git status*

  *On branch master*

  *Untracked files:*

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

  *<filename>*

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

# Git на локальной машине:

Начать отслеживание файла:

- *$ git add <filename>*
- *$ git status*

    *On branch master*

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

        *new file: <filename>*

Сокращенная форма вывода:

- $ git status –s

    M README

    MM Makefile

    A lib/git.rb

    M lib/simplegit.rb

    ?? LICENSE.txt

# Git на локальной машине:

Игнорирование файлов:

- *$ cat .gitignore*

  **.[oa]*

  **~*