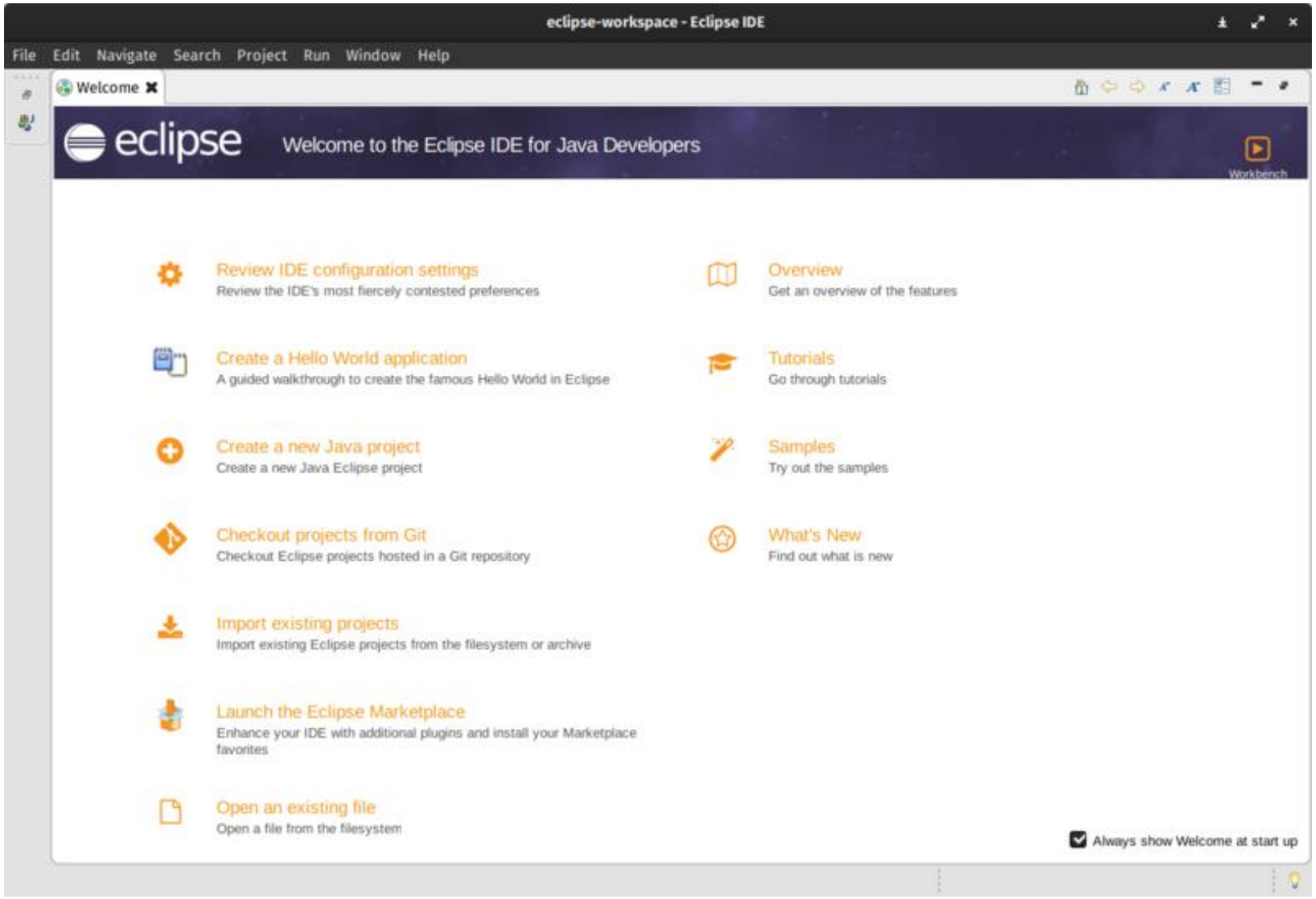
Тема 2. Интегрированная среда разработки ПО

1. Основные компоненты среды разработки
2. Приёмы работы
3. Поддержка совместной работы над проектом

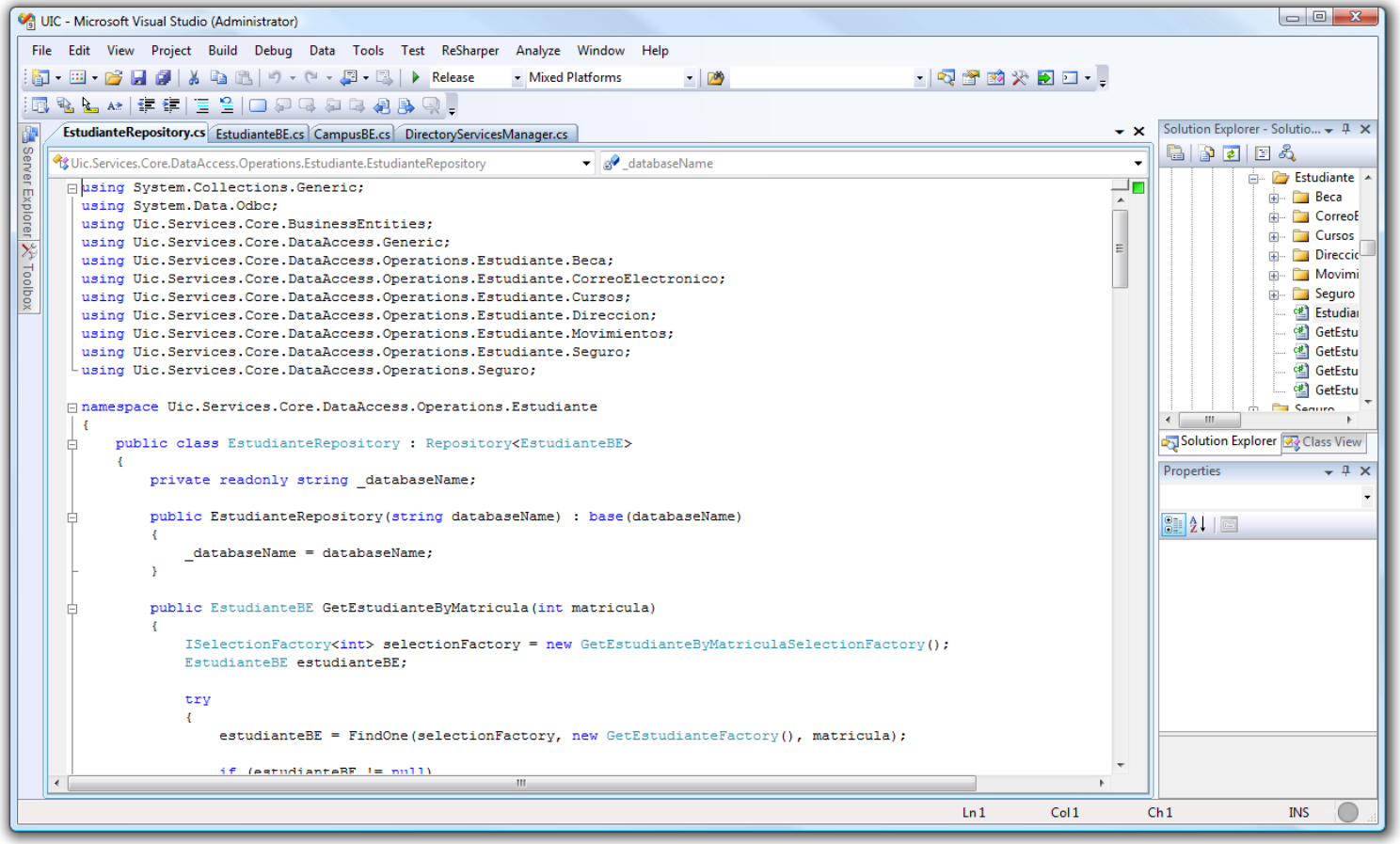
Основные компоненты IDE

- редактор кода
- средства сборки проекта
- средства запуска проекта
- средства отладки
- средства тестирования
- поддержка совместной работы
- расширения

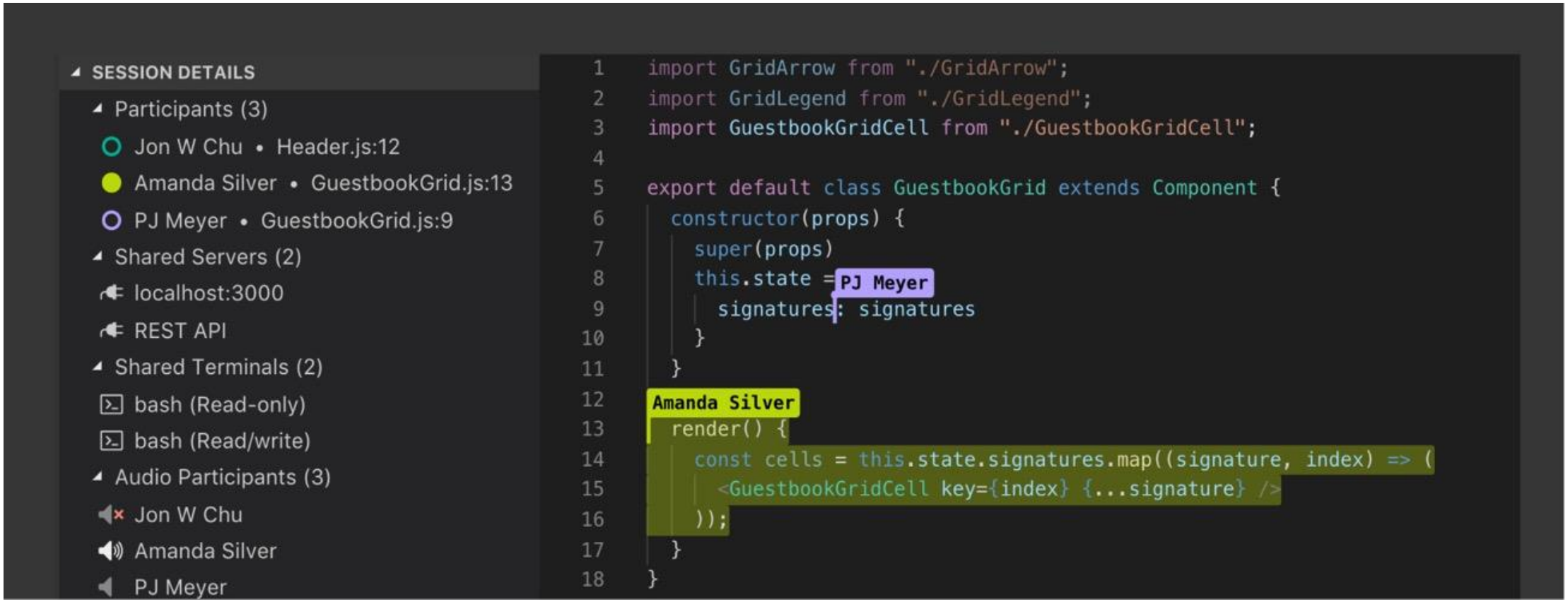
Примеры сред. Eclipse



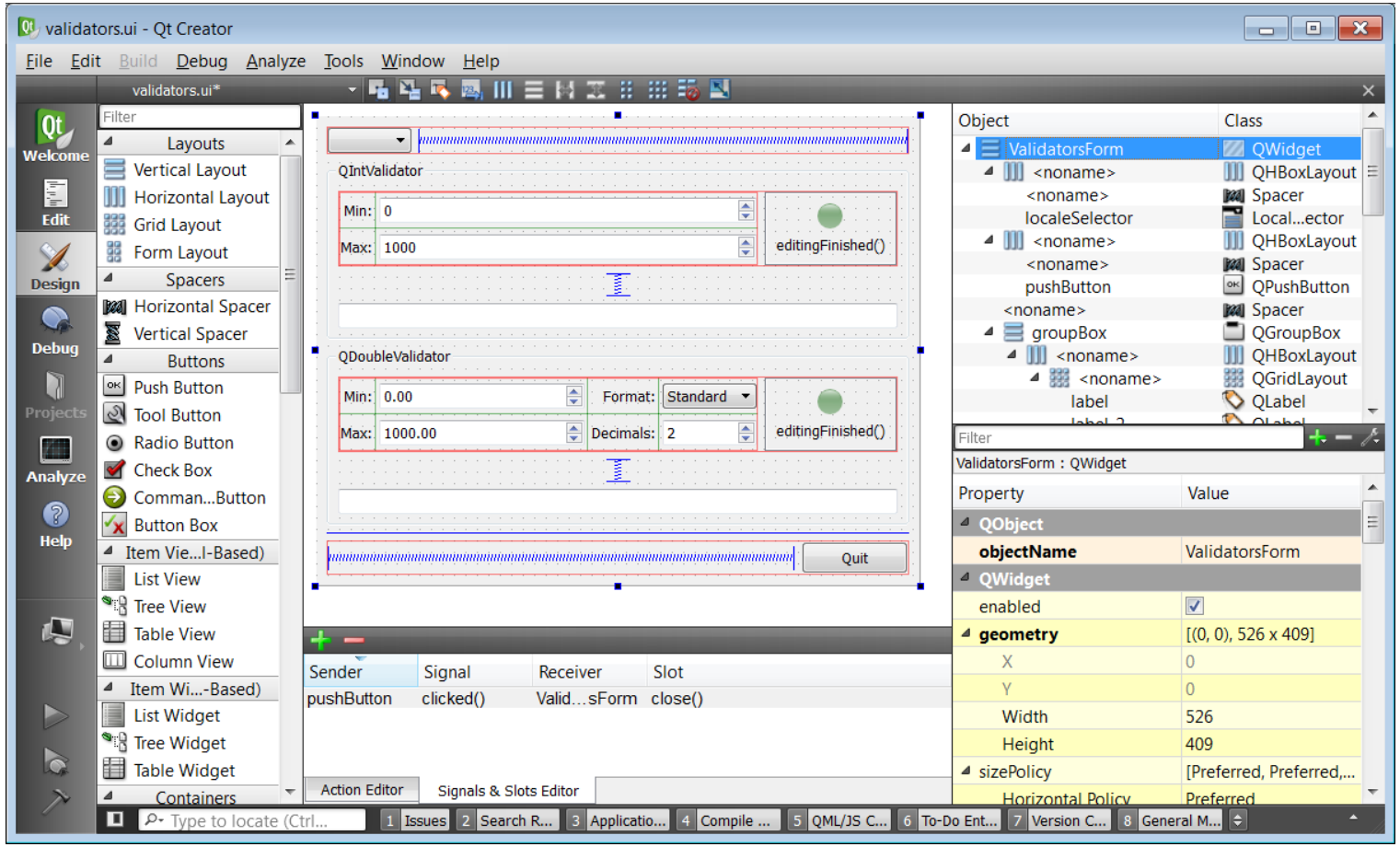
Примеры сред. MS Visual Studio



Live Share в Visual Studio 2019

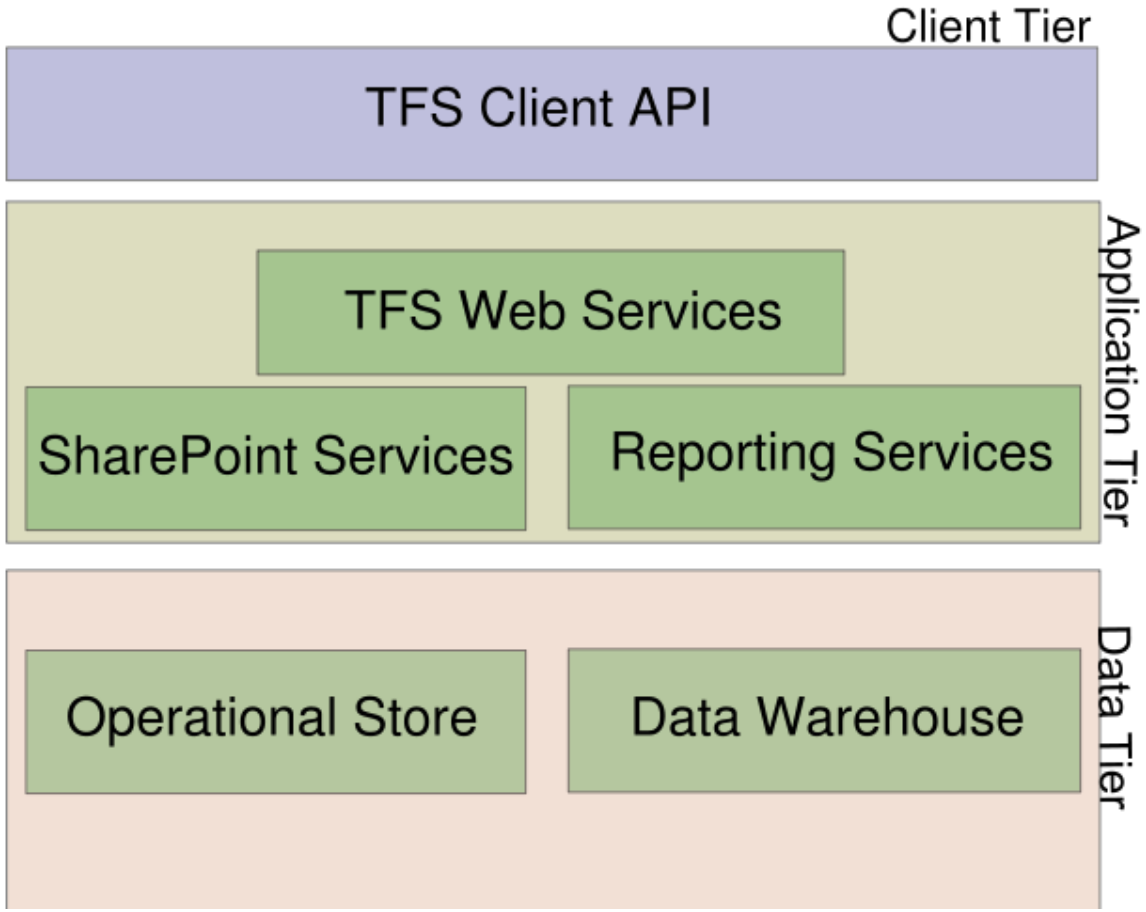


Примеры сред. Qt Creator

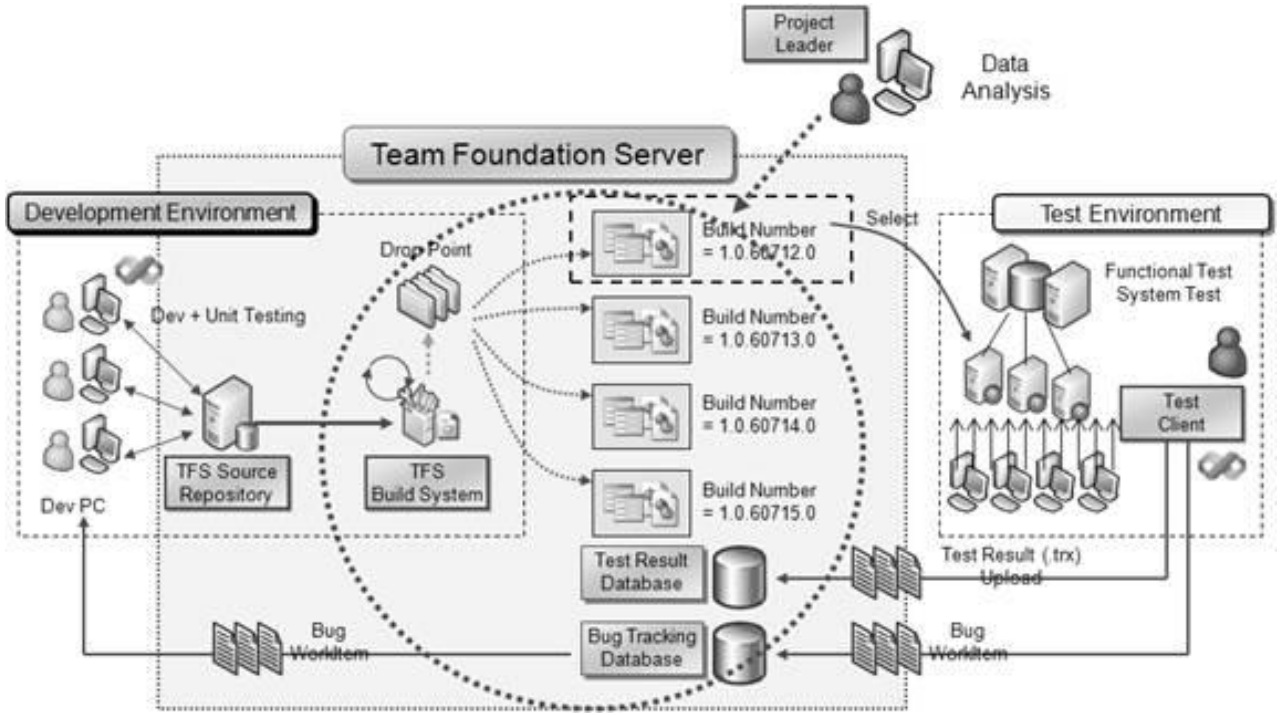


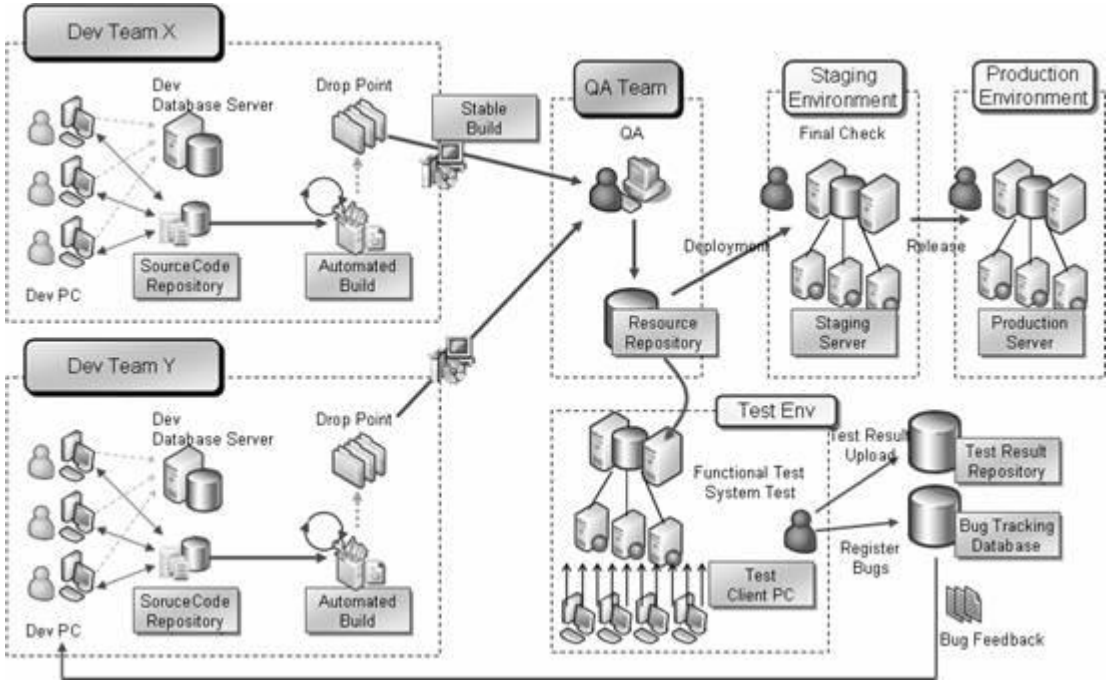
Примеры сред. Azure DevOps Server

Архитектура TFS



Логическая структура





Тема 3. Системы управления версиями

Назначение систем управления версиями

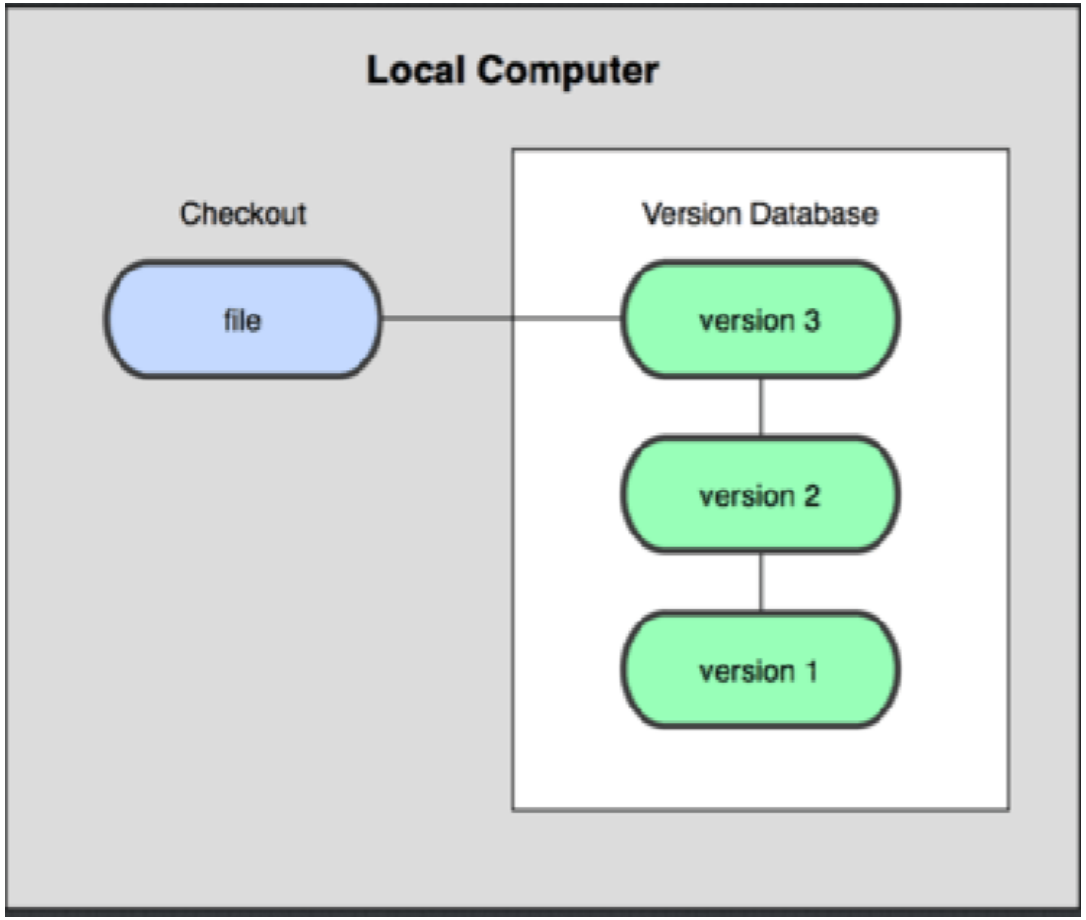
Виды систем управления версиями

Git: основные команды и порядок работы

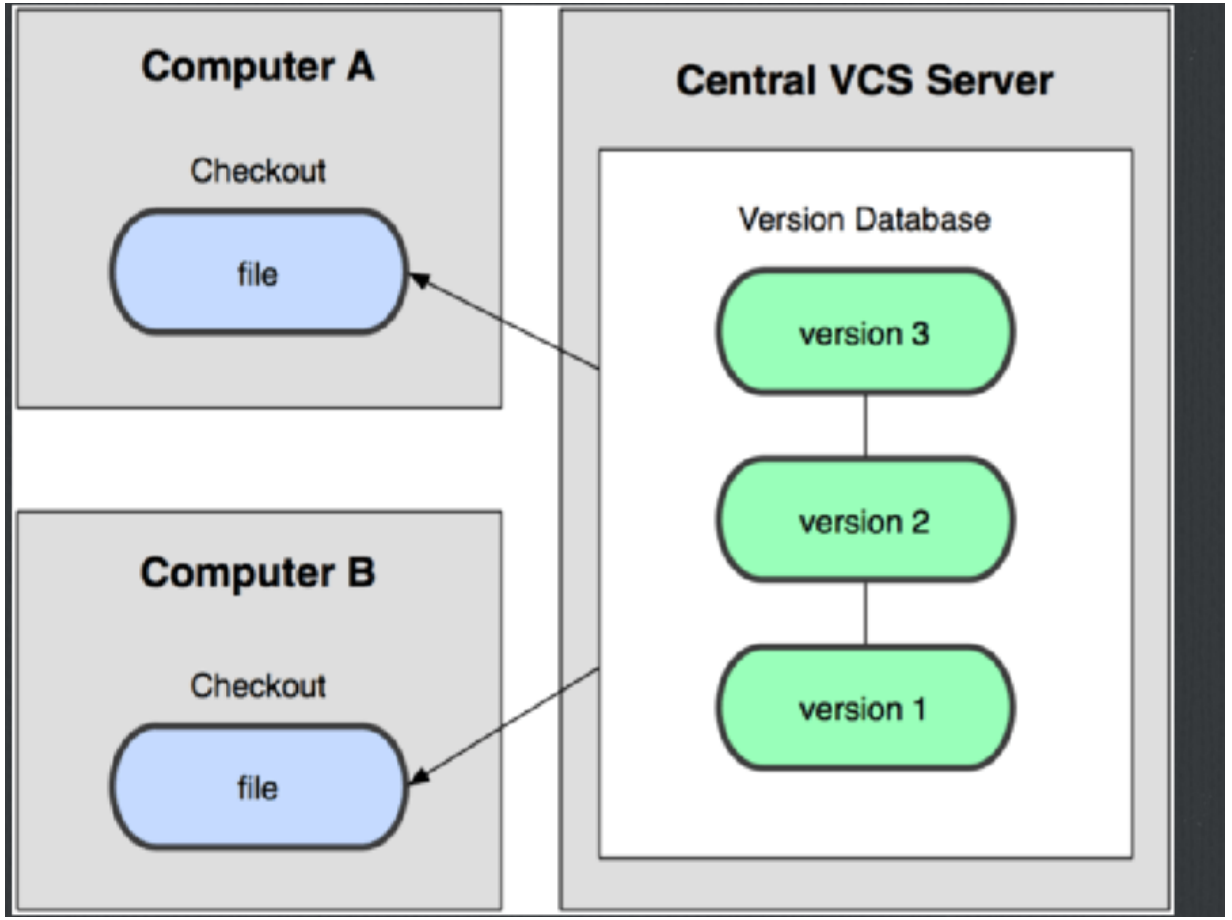
Зачем управлять версиями?

- дисциплина
- совместная работа
- архив всех версий (история)
- восстановление
- экономия дискового пространства

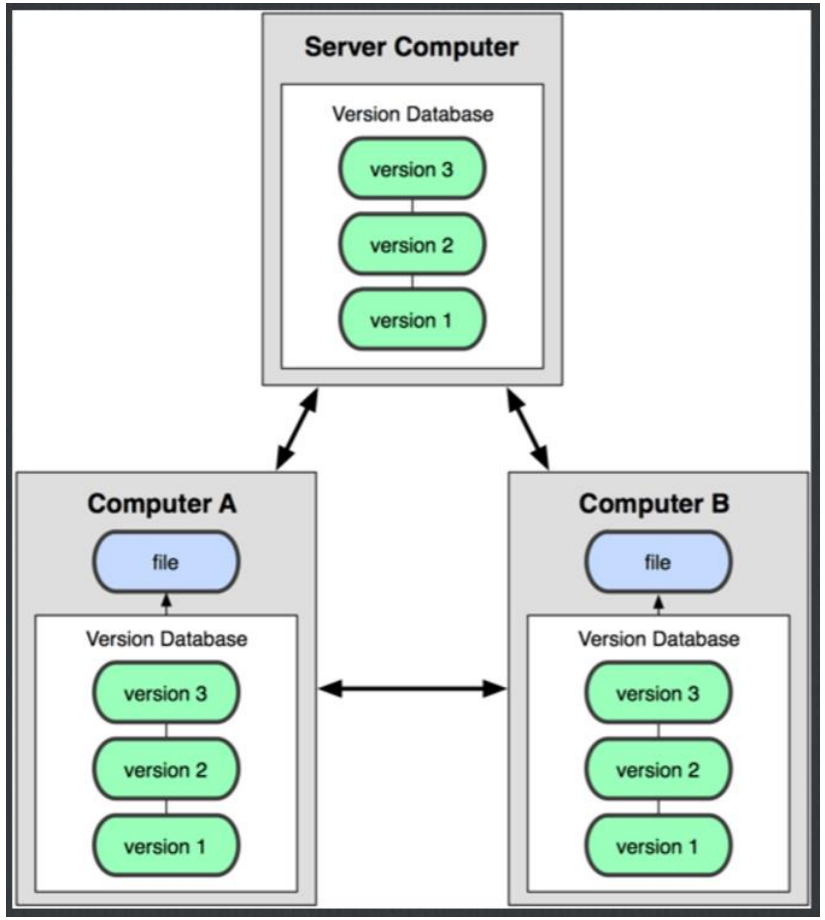
Локальное управление версиями



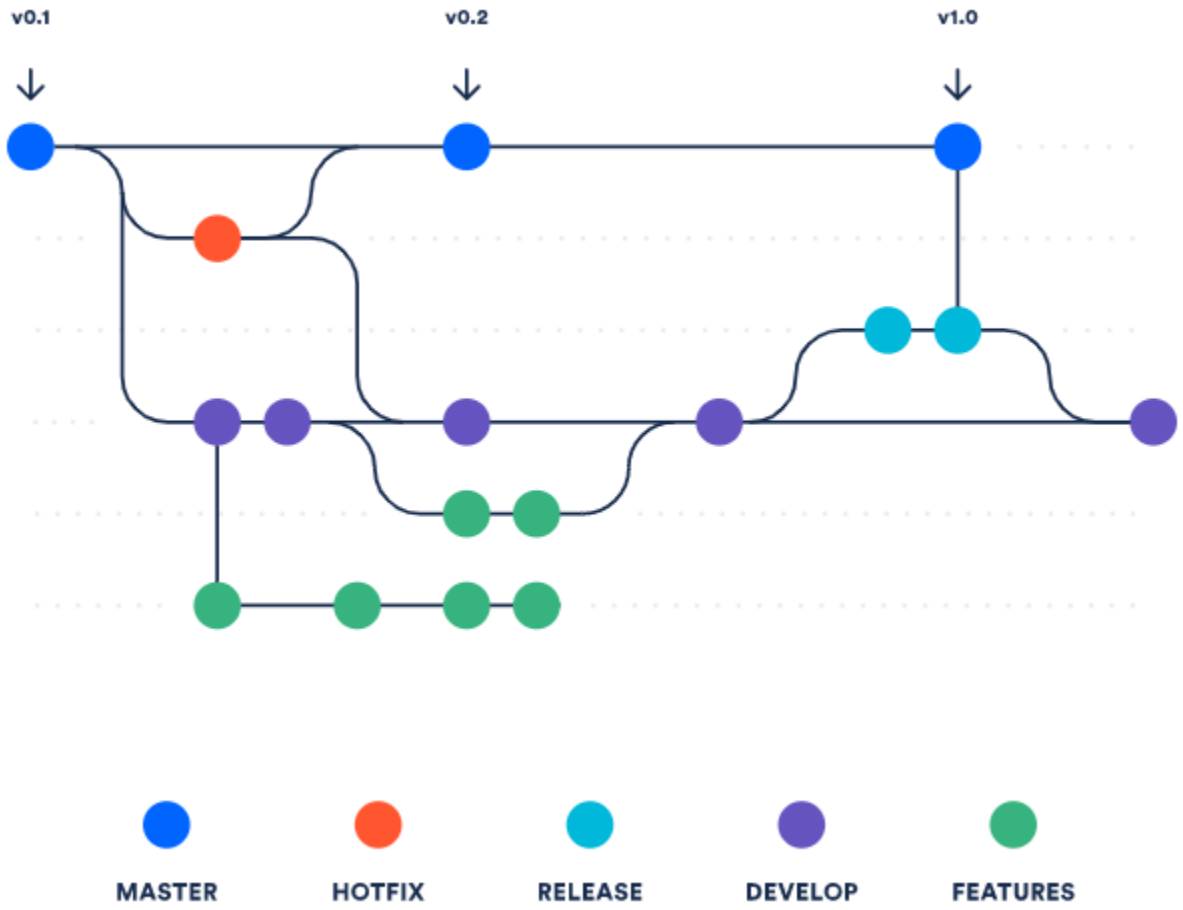
Централизованное управление версиями



Распределённое управление версиями



Как это выглядит



- Система одновременных версий (CVS)
- Apache Subversion (SVN)
- Git
- Mercurial
- Preforce

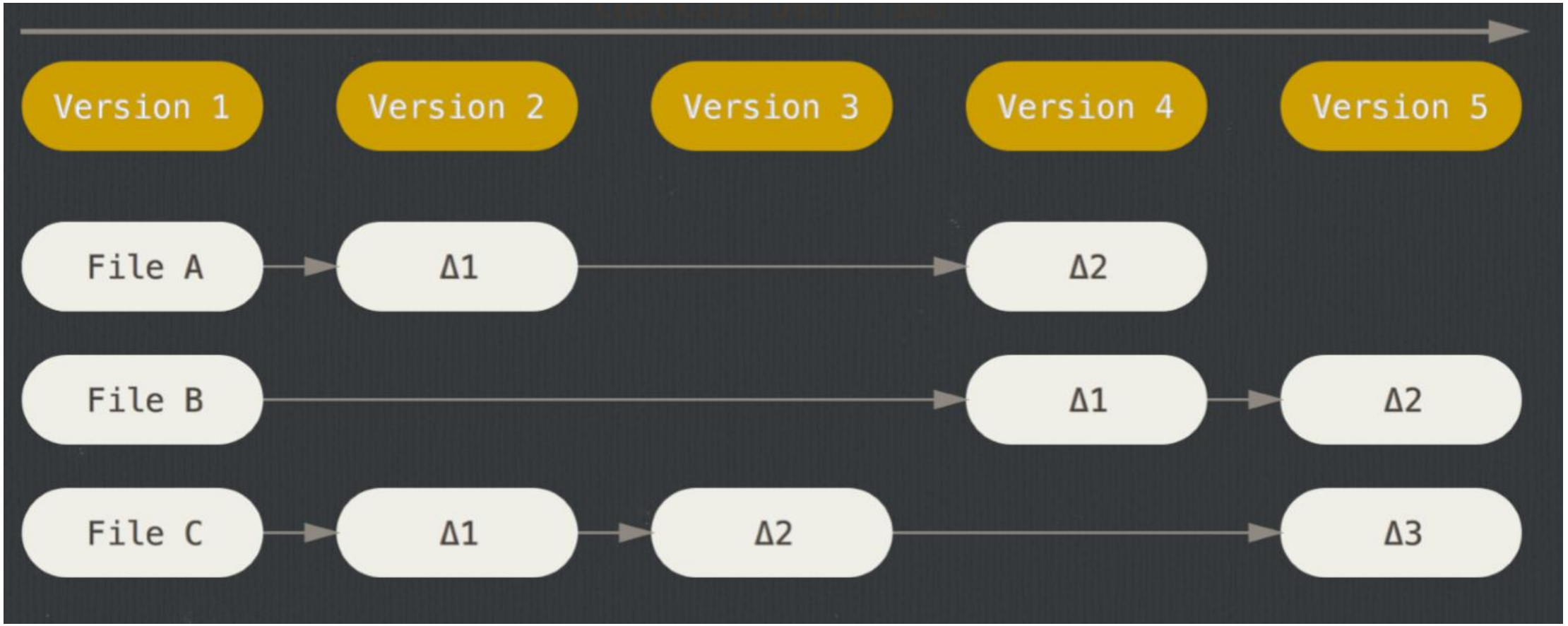
Сравнение версий

Программное обеспечение	Архитектура сети	Разрешение конфликтов	Статус разработки
Git	Распределенная	Слияние	Активная
Mercurial	Распределенная	Слияние	Активная
SVN	Клиент-сервер	Слияние или блокировка	Активная
CVS	Клиент-сервер	Слияние	Только обслуживание

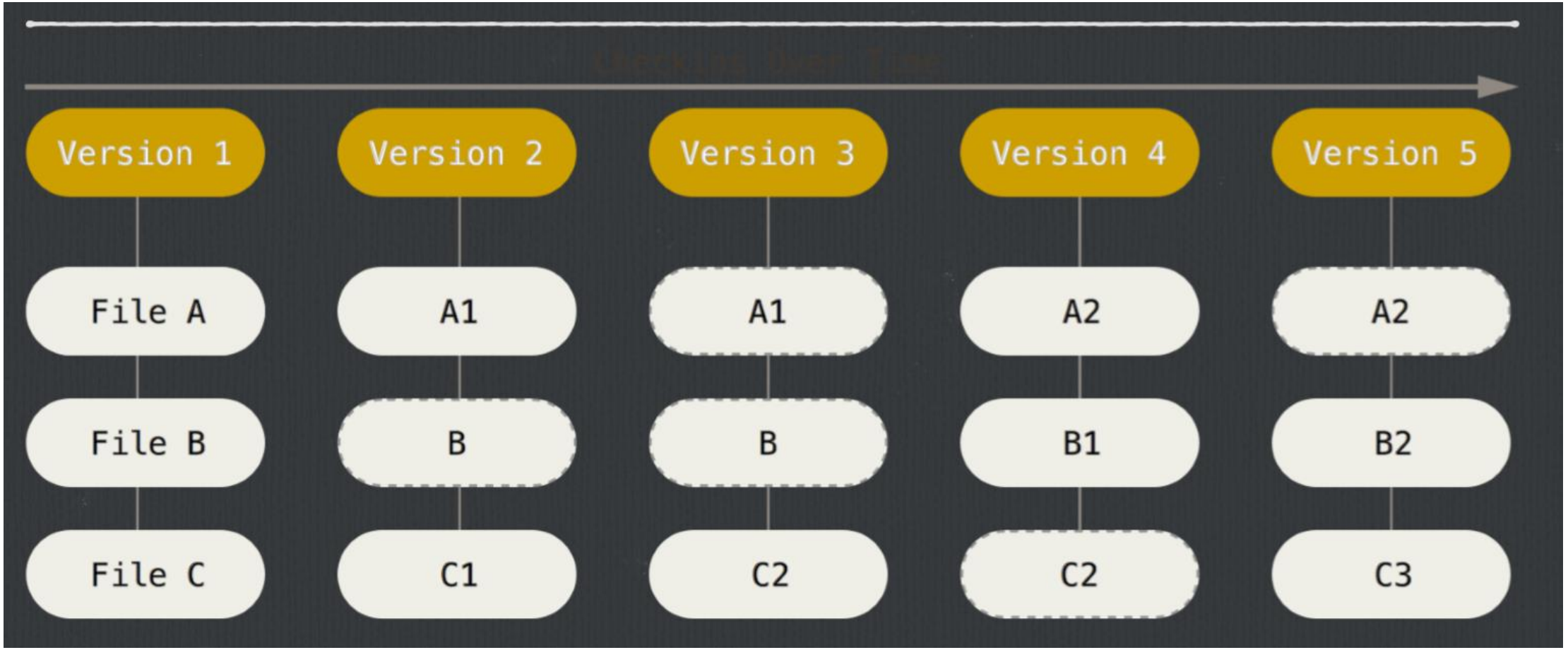
Git

- Скорость
- простой дизайн
- поддержка нелинейной разработки
- Распределённость
- поддержка больших проектов

Хранение изменений



Хранение состояний



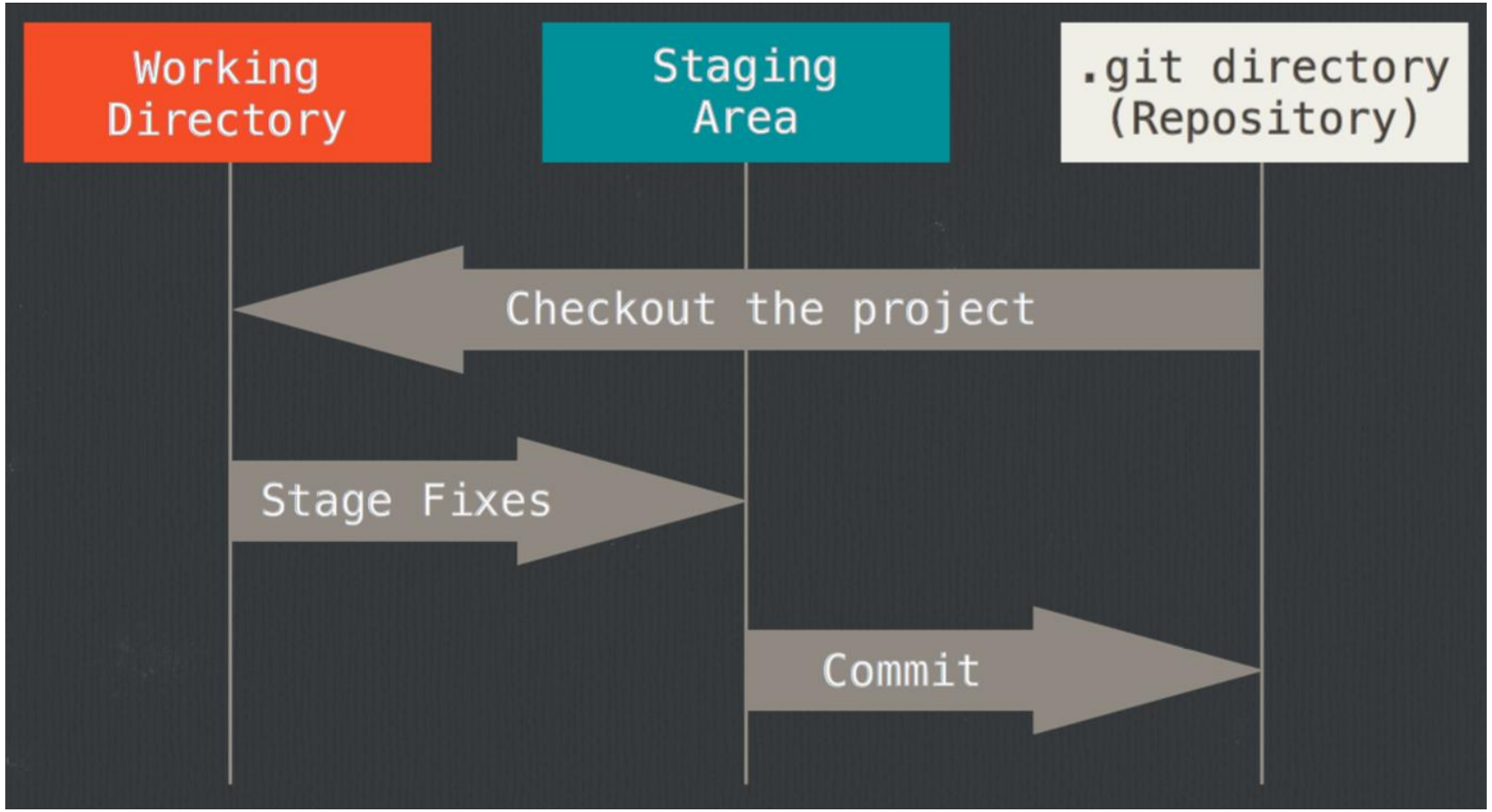
Особенности Git

- локальность изменений
- целостность данных
- добавление данных

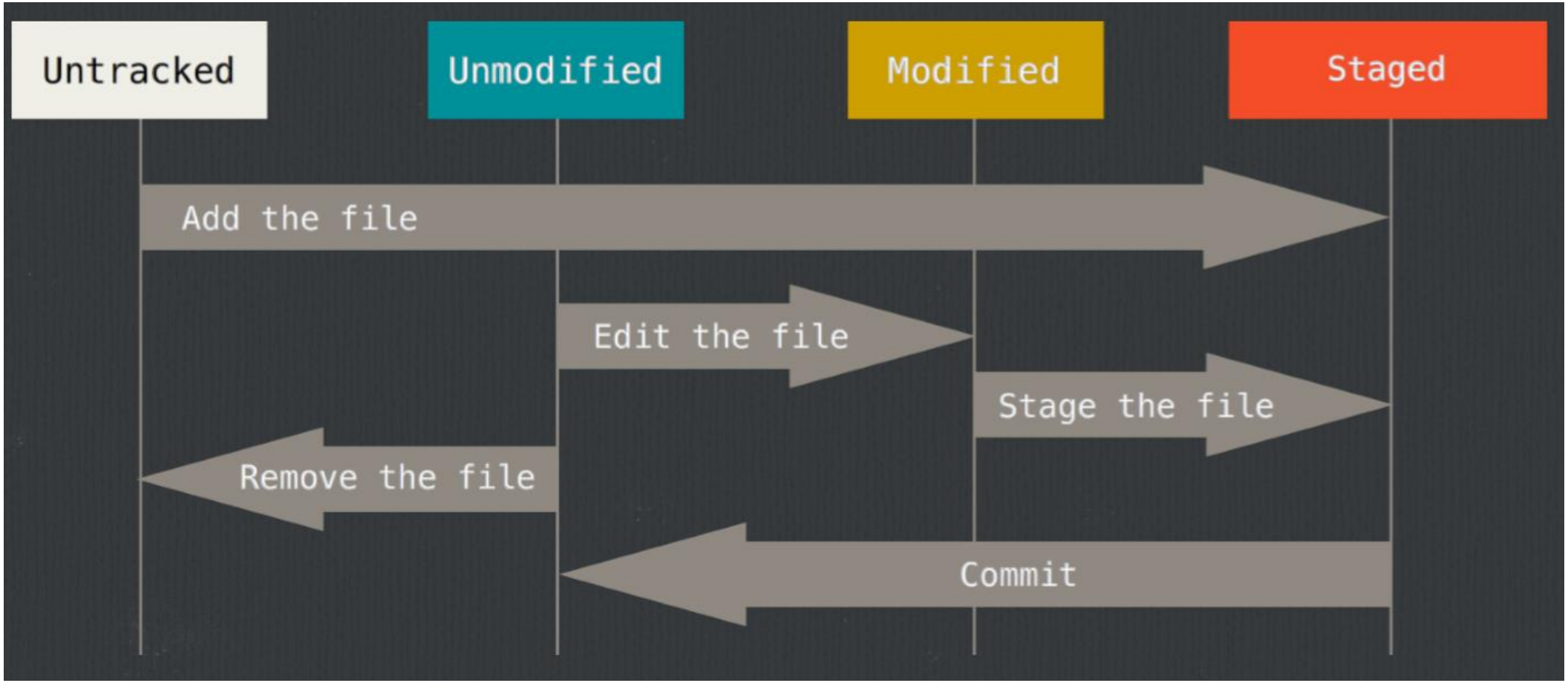
Три состояния файлов проекта

- закреплённое (commited)
- изменённое (modified)
- подготовленное (staged)

Три секции Git-проекта

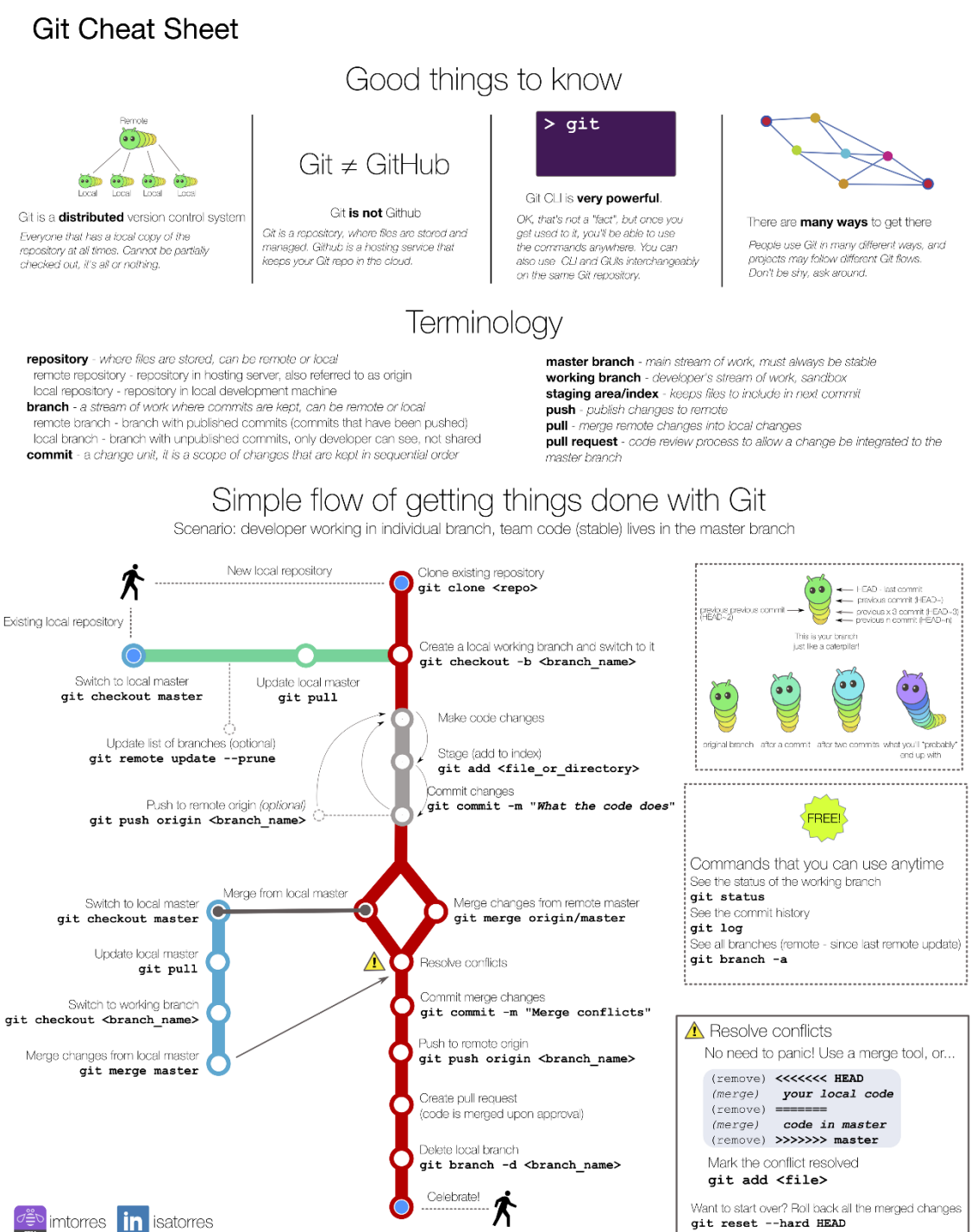


Жизненный цикл состояний файлов проекта



Настройка и работа с Git

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```



Git Cheat Sheet

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

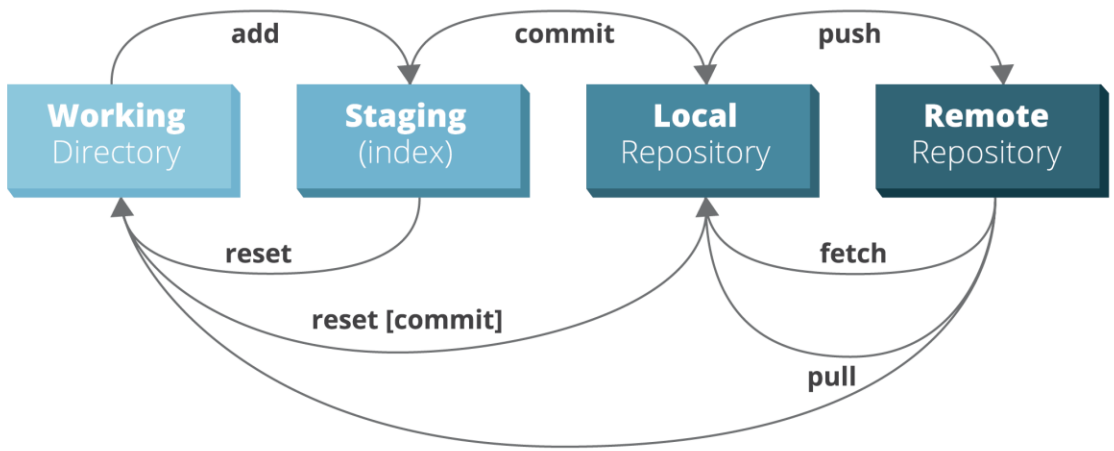
```
$ git push
```

Finally!

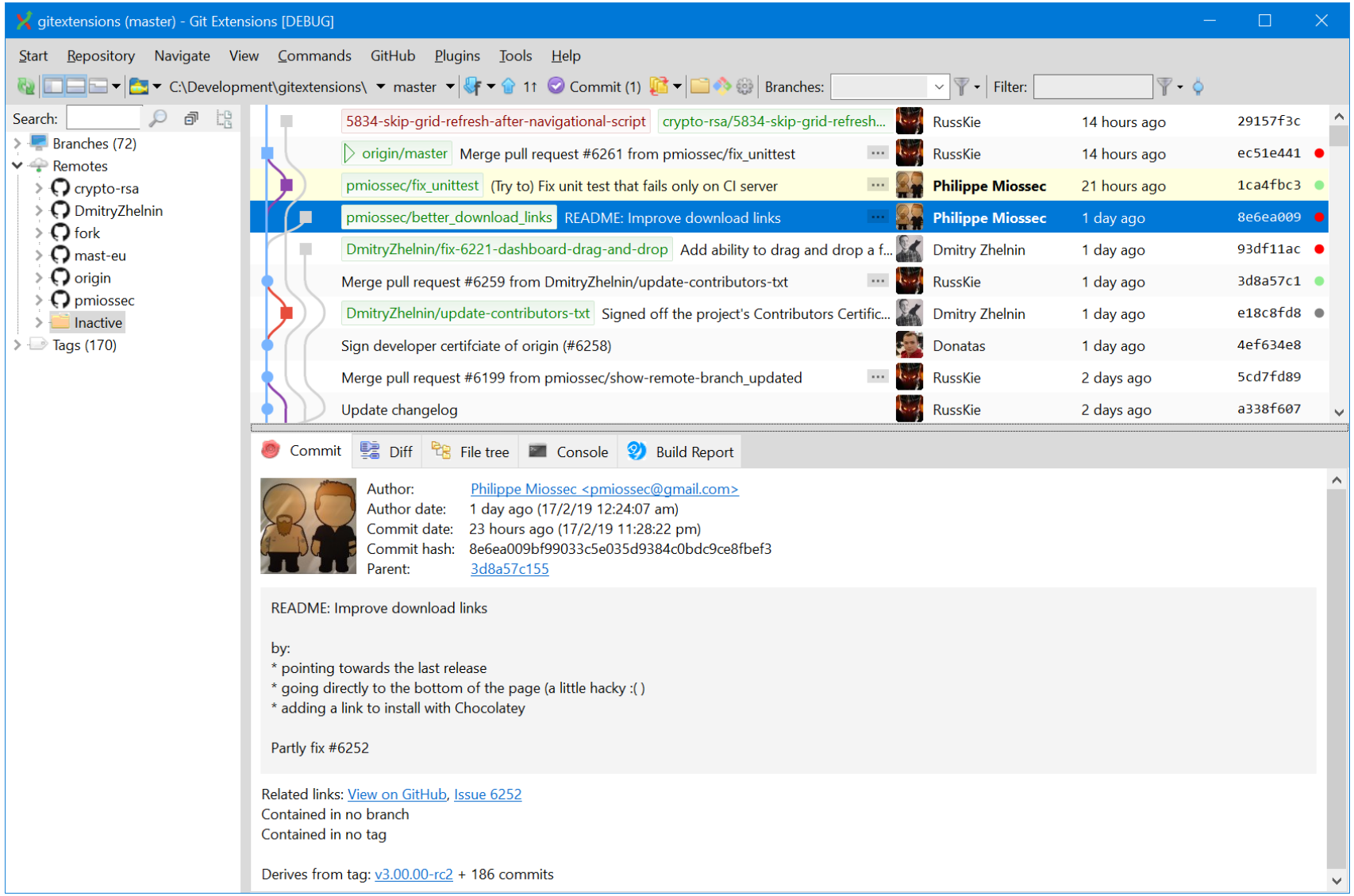
When in doubt, use git help

```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.



Git Extensions



Git Extension