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.

Системы контроля версий

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить.

Классификация систем контроля версий

- 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.

Ежедневный цикл работы

1. Обновление рабочей копии.

Разработчик выполняет операцию обновления рабочей копии (update) насколько возможно

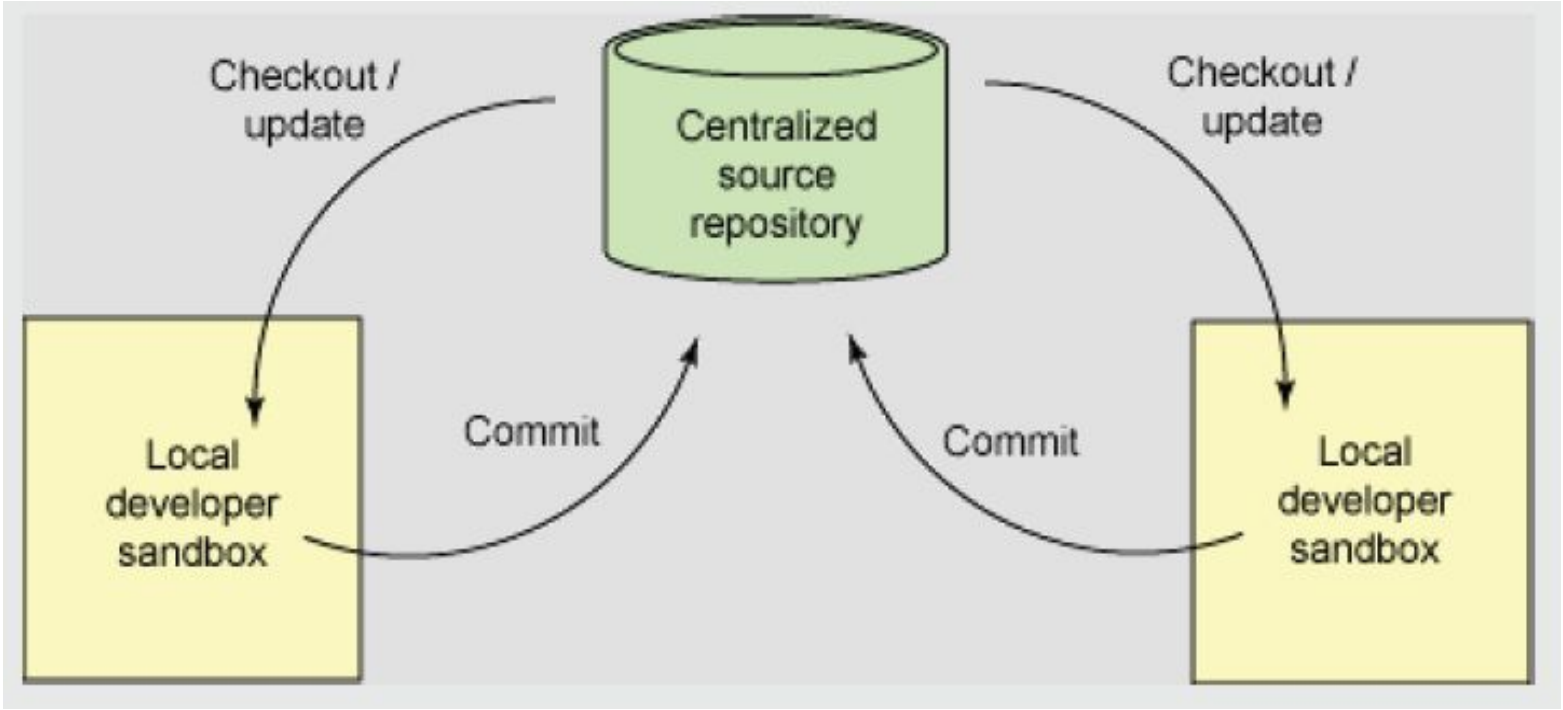
2. Модификация проекта.

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

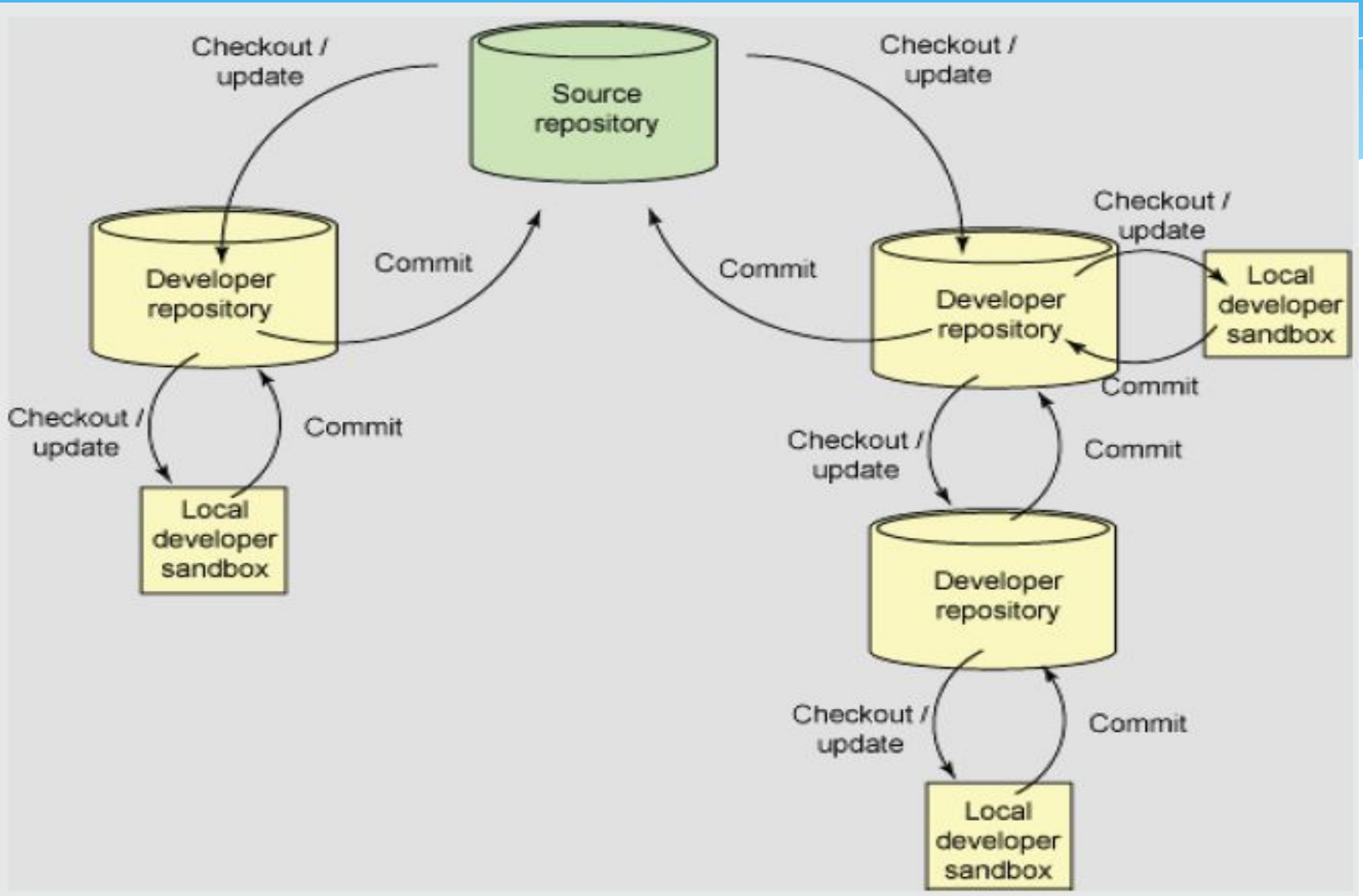
3. Фиксация изменений.

Завершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер.

Централизованные VCS



Распределённые VCS



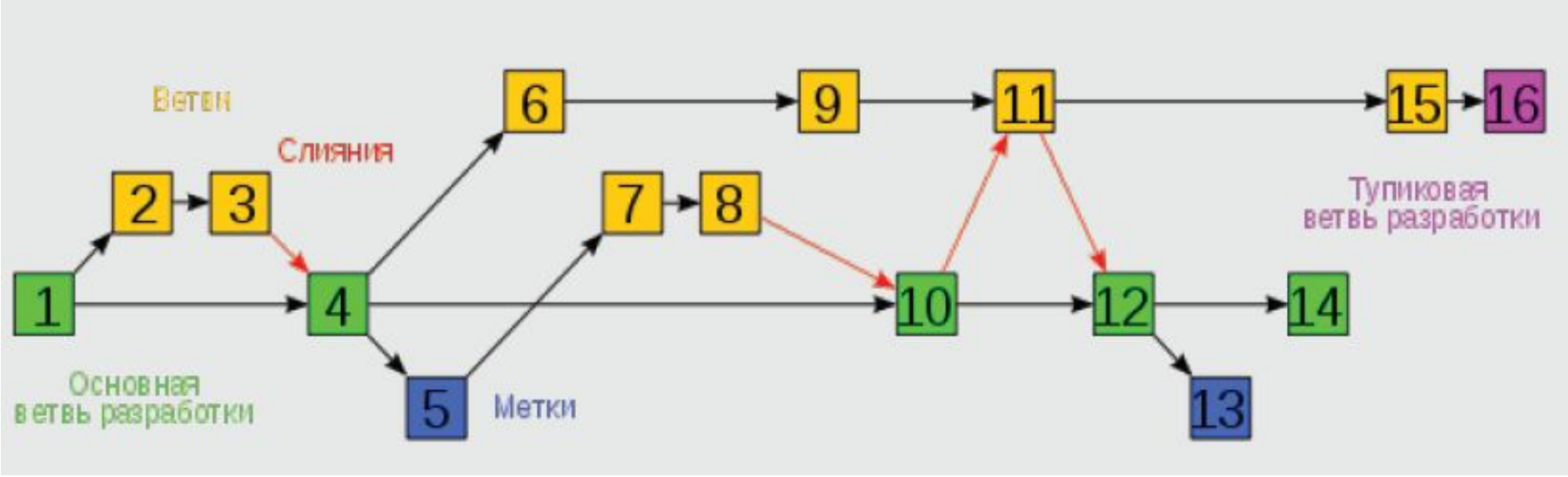
Основные термины

- **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.
- **tag, label** — метка, которую можно присвоить определённой версии документа.

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 эволюции ветвей в проекте



GIT

Git — распределённая система управления версиями файлов.

Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux в 2005 году.

Плюсы:

+ Высокая производительность.

+ Развитые средства интеграции с другими VCS.

+ Репозитории git могут распространяться и обновляться общесистемными файловыми утилитами, такими как rsync.

Схема работы с Git

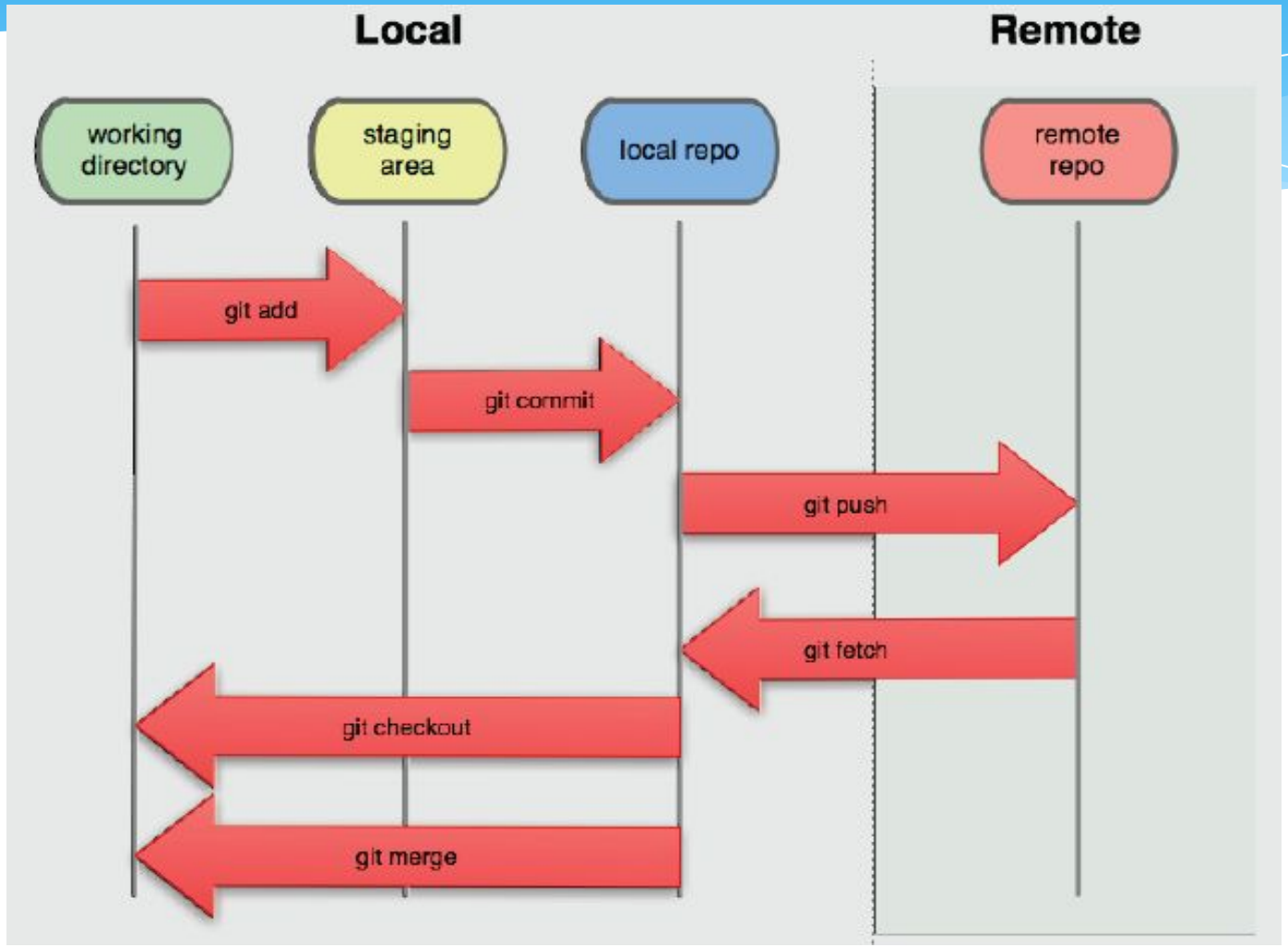


Схема работы с Git

git pull/ fetch — забираем изменения из центрального репозитория

git push — вносим изменения в удаленный репозиторий

git commit — совершение коммита

git checkout — переключение между ветками, извлечение файлов

git merge — слияние веток (разрешение возможных конфликтов)

git add — позволяет внести в индекс— изменения, которые затем войдут в коммит