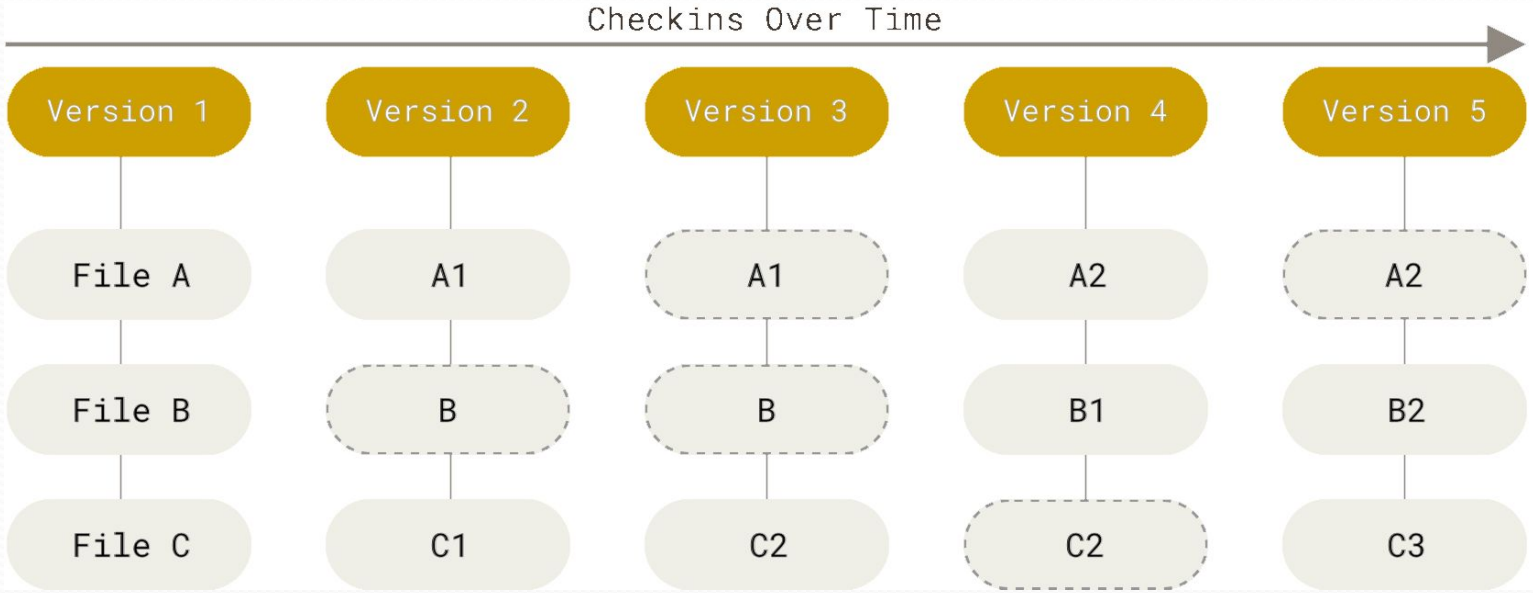
Гунба С.В.

Система контроля версий GIT

О системе контроля версий Git

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

Git сохраняет файлы разных версий как «снимки». Если в новой версии какой-либо файл не менялся, то сохраняется не «снимок», а ссылка на предыдущую версию файла.



Основные понятия GIT

- Репозиторий – база данных где хранится информация о файлах («снимках»),история изменений, ссылки на файлы, комментарии и тд.

Операции над репозиториями:

1. Создание. Создать репозиторий можно в любой папке не находящейся под версионным конролем
2. Клонирование. Можно клонировать любой репозиторий. При этом данные находящиеся в исходном репозитории будут скопинованны в новый репозиторий

Все операции в GIT осуществляются с помощью команд из командной строки. Но к счастью есть оболочки с графическим интерфейсом, которые облегчают работу.

Построение работы с репозиториями (вариант)

В главный репозиторий загружается готовый функционал, при этом в него попадают так же «промежуточные версии ПО»

Git

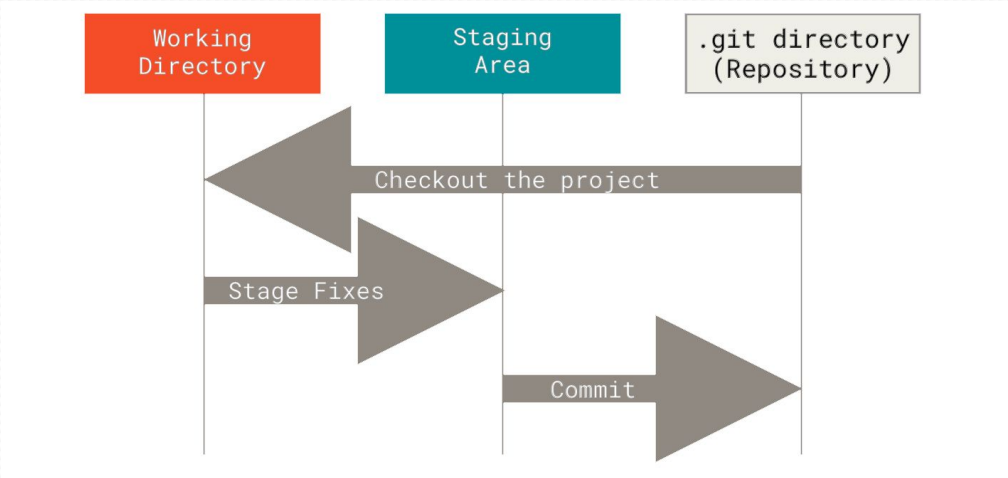
Файл для Git могут быть в 3 основных состояниях:

Изменен (modified),Индексирован (staged), Зафиксирован (Committed).

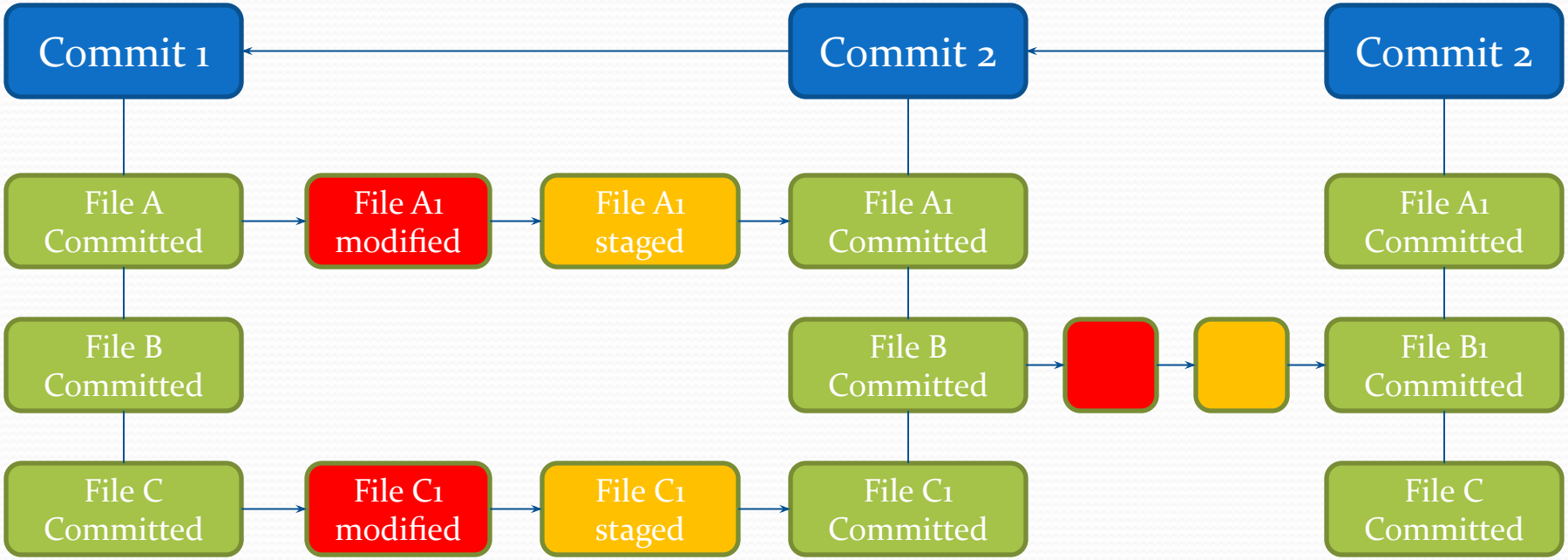
К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы.

Индексированый — это изменённый файл в его текущей версии, отмеченный для включения в следующий коммит.

Зафиксированный значит, что файл уже сохранён в вашей локальной базе.



Подробнее о состояниях файлов



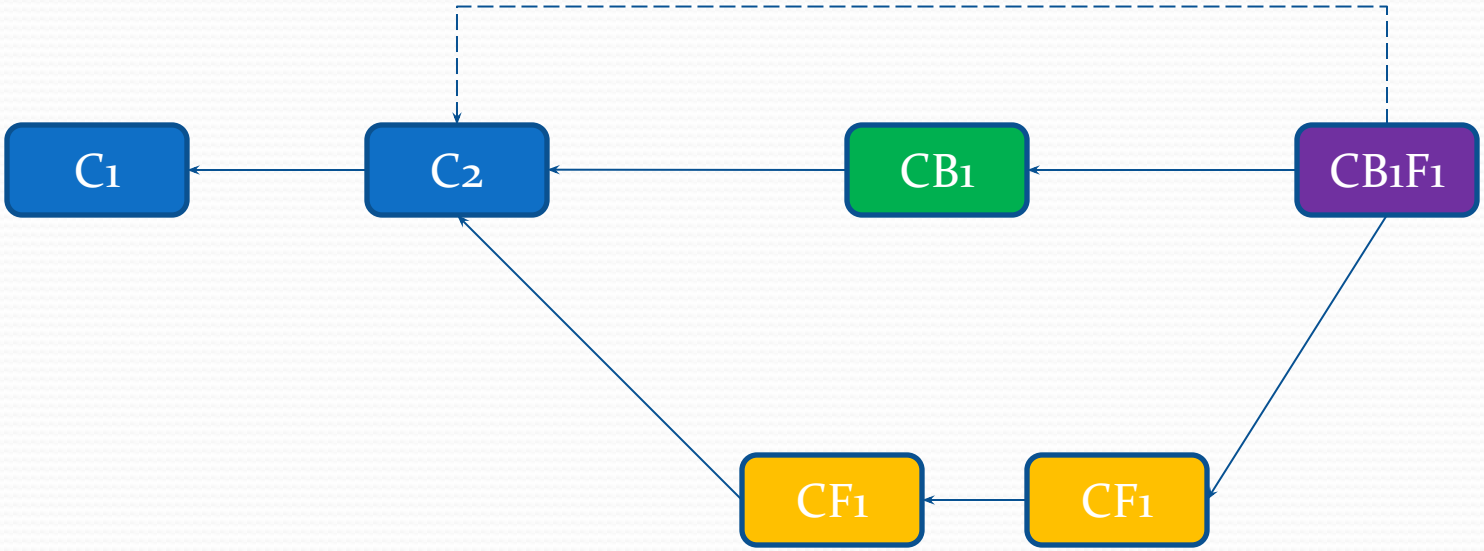
Термины работы с файлами и репозиторием

- Clone – клонирование репозитория
- Init – создание репозитория

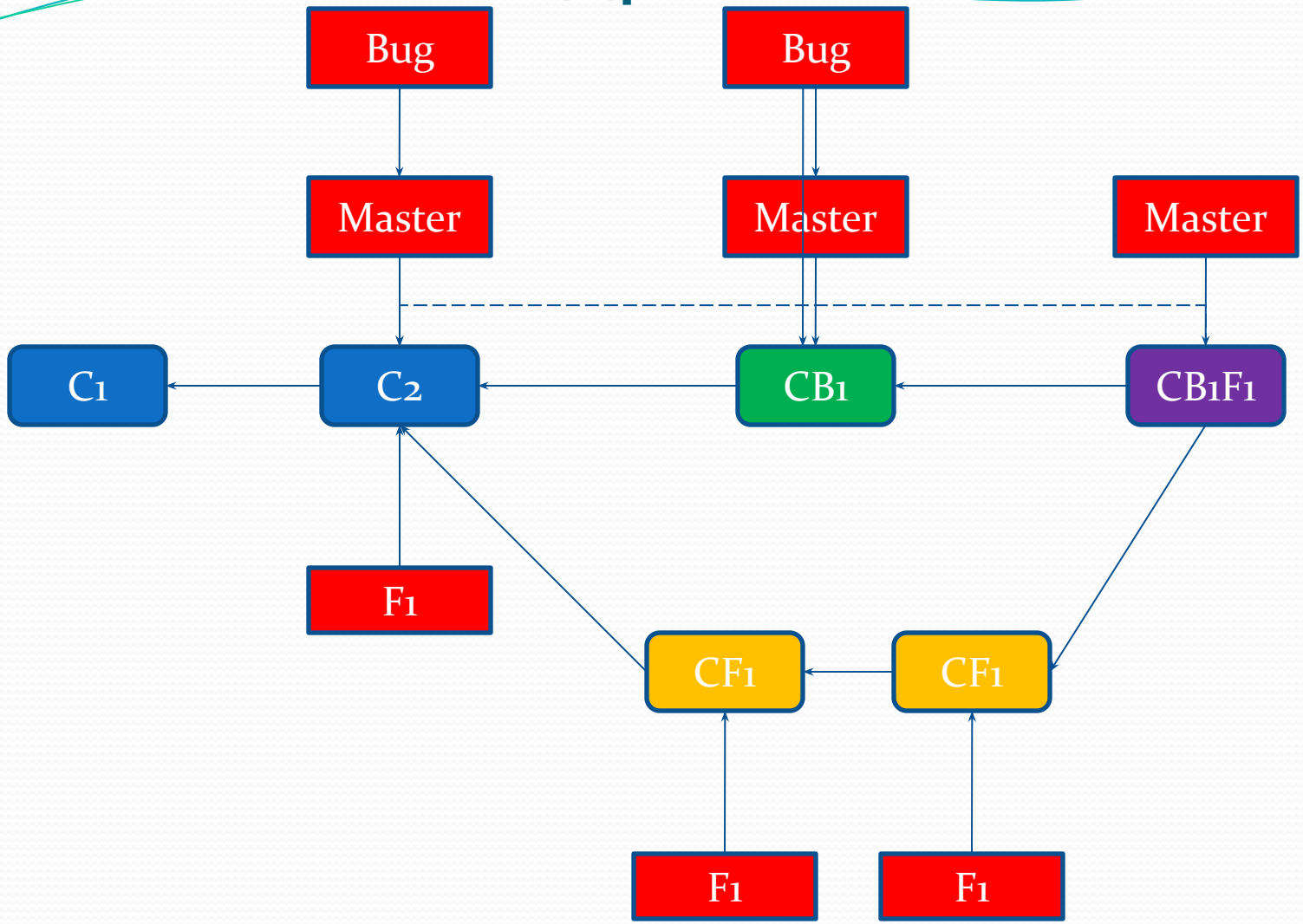
- Add – команда добавить файл в индекс
- Commit – команда фиксации изменений в репозитории (создание «снимка» файлов)

При Commit необходимо указывать комментарий, описывающий что мы коммитим.

Ветвление в Git



Ветвление подробно



Слияние

Слияние бывает 2х типов:

1. Быстрое перемещение (Fast-Forward)
2. Коммит слияния (слияние имеет более одного предка)

Быстрое перемещение-это прямое слияние с предыдущим коммитом (коммит с одним предком). По сути просто перемещение указателя ветки. Как правило проходит без проблем (Слияние Mastrer и Bug).

Коммит слияния сливает изменения из разных веток. При таком подходе могут возникать конфликты.

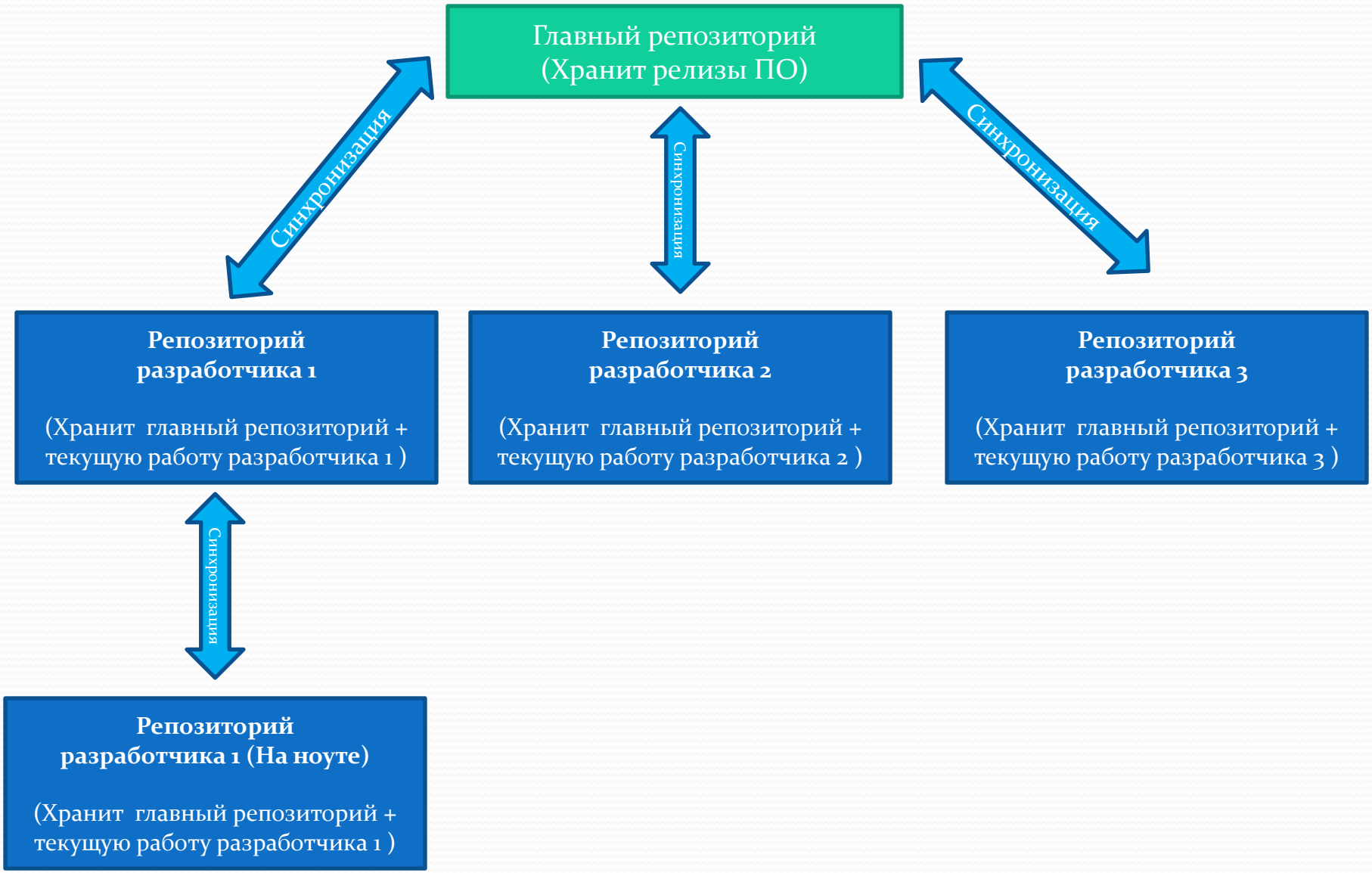
При конфликте Git выдает отчет (пример):

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

Термины ветвления

- Head – Указатель на текущую рабочую ветку
- Branch – команда создание новой ветки, но Head указывает на старую ветку
- Checkout – команда переключения ветки (перенос Head на другую ветку)
- Merge – команда слияния активной ветки (на которую указывает Head) и другой ветки (указывается пользователем).

Работа с удаленными репозиториями



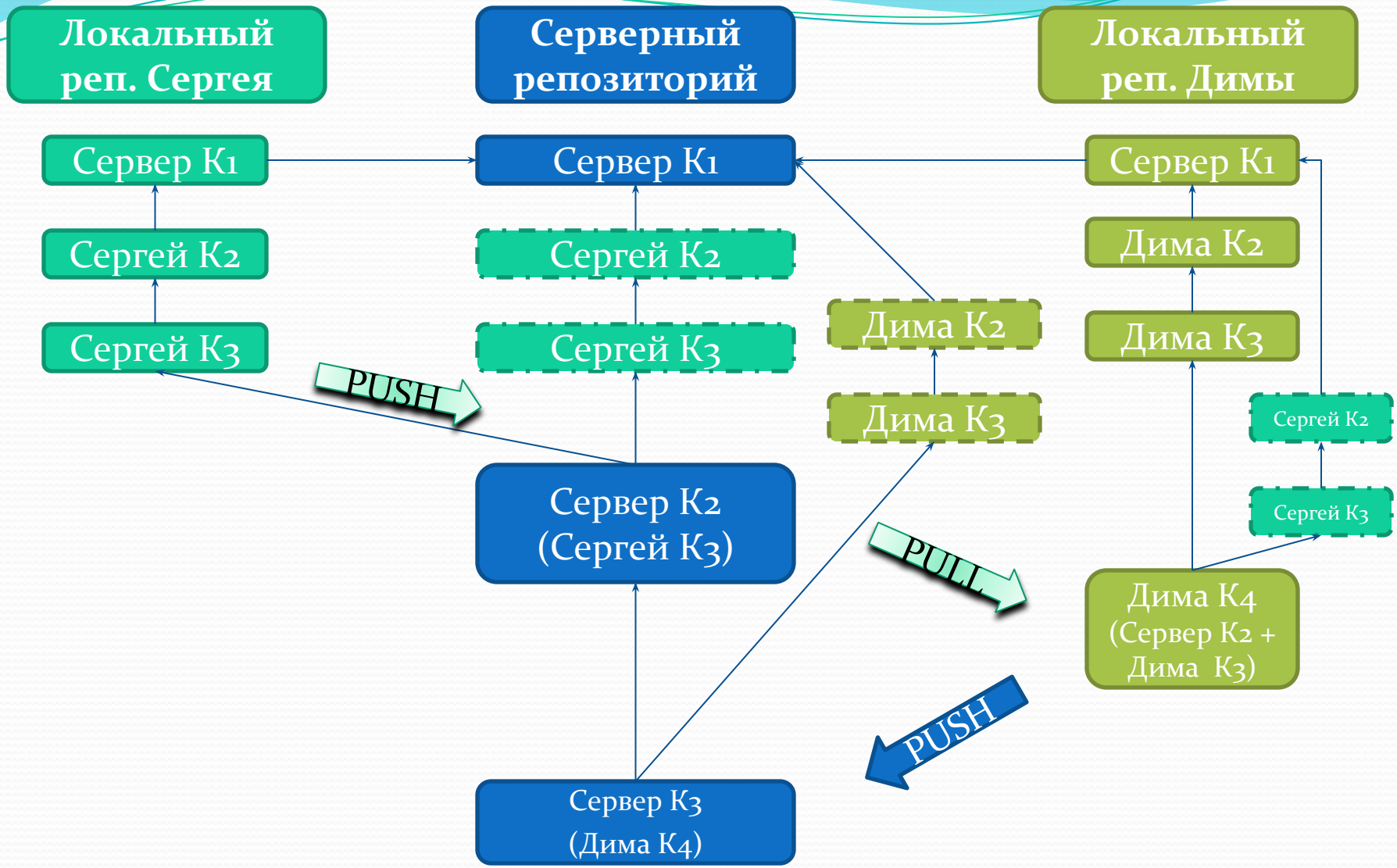
Команды работы с удаленными репозиториями

Fetch – забирает данные в локальный репозиторий, но не сливает их с текущими наработками. Не модифицирует то над чем сейчас работаешь.

Pull – забирает изменения из удаленной ветки и автоматически сливает их с **текущей** веткой.

Push – отправка данных на сервер и автоматическое слияние с выбранной веткой.

Совместная работа



Решение конфликтов

При 3х стороннем слиянии возможно возникновение конфликтов. Конфликты возникают когда в одни и те же строки изменяются разными авторами.

При возникновении конфликтов процесс слияния останавливается, выдается предупреждение и список файлов в которых есть конфликты.

В файлах конфликты выделяются следующим образом:

Текст файла	Описание
<<<<<<< HEAD Дима добавил еще а Сергей тоже добавил в ту же строку ======= Дима добавил еще и еще >>>>>> 47f6294b502b59f6dfe1ac3a11f96d733428ded6	<<<Метка начала конфликта Версия в Head ветке === Разделитель Версия в «внешней» ветке >>> конец разделителя

После разрешения конфликта необходиом произвести коммит.

Командная работа подробнее

Работа с серверным репозиторием состоит из следующих шагов:

1. Локальный коммит инкремента (нового функционала)
2. Запрос изменений с сервера (Fetch)
3. Если они есть, то забираем измненния и сливаем со своей веткой (Pull)
4. Разбираемся с конфликтами, если они есть
5. Тестируем функционал версии с результатами слияния!
6. После прохождения тестов отправляем на сервер (Pull)
7. Новый функционал в общем репозитории!!!

Что храним в репозитории

1. Исходный код ПО
2. Проект ПО с относительными путями (если это возможно)
3. Вспомогательное ПО и библиотеки (ipserial.dll, формирователь *.nrn, терминалы и т.д.)
4. Документацию (требования, протоколы обмена, тест планы)
5. Средства разработки (компилятор, среда разработки и тд)
6. Проекты DipTrace
7. И т.д.