

# Система контроля версий Git

# НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

# ЦЕЛЬ

Получить представление о системе контроля версий Git,  
научиться применять Git на практике

# ПЛАН ЗАНЯТИЯ

СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ

git

UNIX-команды

Работа с репозиторием

Коммиты

# СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ

**Система контроля версий** - инструмент для совместной работы над исходным кодом.

Основные аспекты:

Отслеживание изменений

Сотрудничество

Версионирование

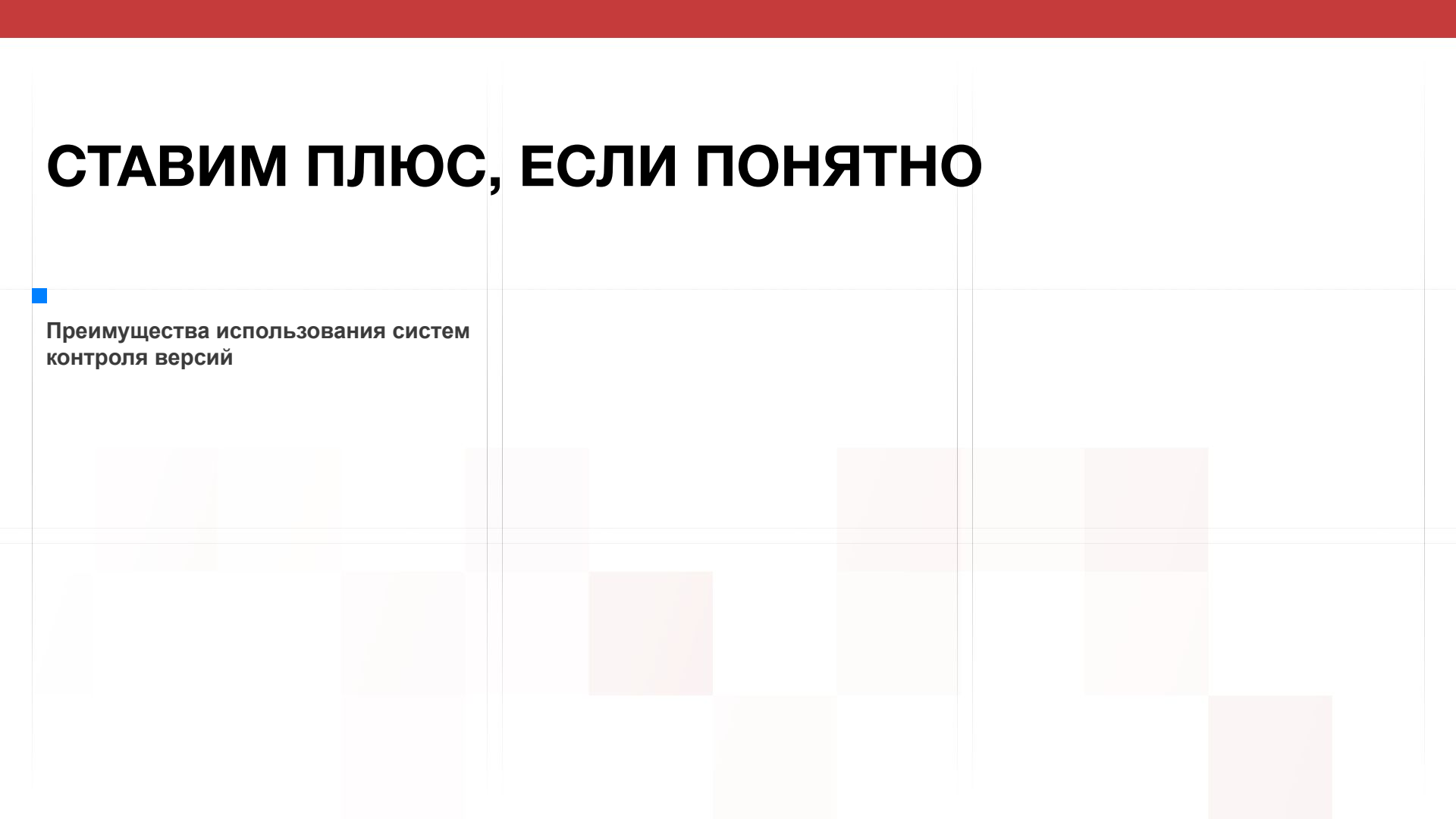
Ветвление и слияние



git



# СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Преимущества использования систем  
контроля версий

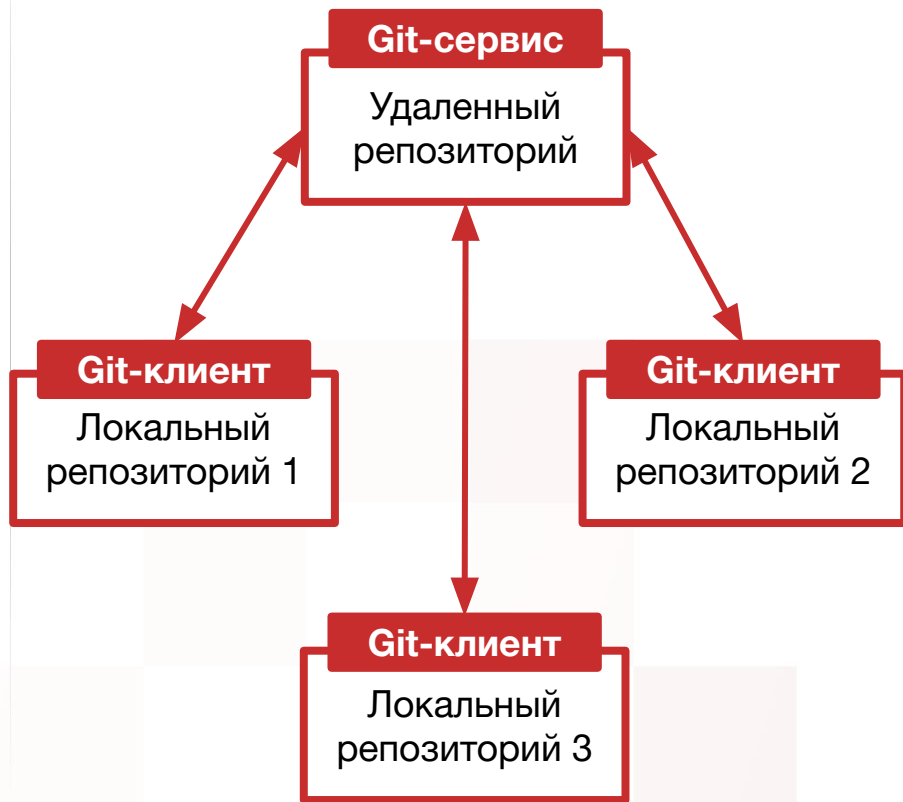
# Git

**Git** - наиболее популярная распределенная система контроля версий. Использовалась при разработке ядра Linux. Распределенность позволяет каждому программисту хранить полную **локальную** копию **удаленного репозитория**, периодически выполняя синхронизацию.

**Репозиторий** - место для хранения кода. Содержит актуальную версию всех файлов и их полную историю изменений.

**Git-сервисы** - предоставляют возможность хранения удаленных репозиториях и графические интерфейсы для работы. Наиболее популярные - **Github.com**, **Gitlab.com**, **Bitbucket.com**

**Git-клиент** - приложение, используемое для взаимодействия с Git-сервисом и репозиториями в нем. Клиентом может быть Bash, GUI или IDE.



# СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Репозиторий

Распределенная система контроля  
версий

Git-сервис и git-клиент



# Unix-команды

Переход в папку:

**cd** ПУТЬ

Просмотр содержимого текущей папки:

**ls**

Переход на уровень выше:

**cd ..**

Создание папки:

**mkdir** НАЗВАНИЕ\_ПАПКИ

Создание файла:

**touch** НАЗВАНИЕ\_ФАЙЛА

Удаление файла

**rm** НАЗВАНИЕ\_ФАЙЛА

Удаление папки

**rm -rf** НАЗВАНИЕ\_ПАПКИ

# СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Использование Linux-команд в Git  
Bash

# РАБОТА С РЕПОЗИТОРИЕМ (Git Bash)

**Клонирование** (clone) - операция, позволяющая создать локальную копию удаленного репозитория. Для клонирования репозитория следует выполнить команду:

**git clone URL-репозитория**

Далее, можно перейти в папку с нужным репозиторием с помощью команды **cd** и проверить статус локального репозитория:

**git status**

В репозитории не должны сохраняться скомпилированные файлы или служебные файлы проекта. Необходимо в репозитории создать файл **.gitignore** и указать расширения и имена файлов, которые мы не хотим видеть в репозитории

Теперь с помощью команды **git status** мы видим, что файл **.gitignore** не отслеживается Git-ом. Необходимо добавить его в индекс:

**git add .gitignore**

Сейчас **git status** показывает, что он видит изменения (новый файл **.gitignore**) и их можно зафиксировать (сделать **коммит**)

**git commit -m 'add gitignore'**

Отправим изменения в удаленный репозиторий

**git push**

Про то как вернуться к старым версиям кода: подробно [ТУТ](#)

# СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

■ Создание репозитория

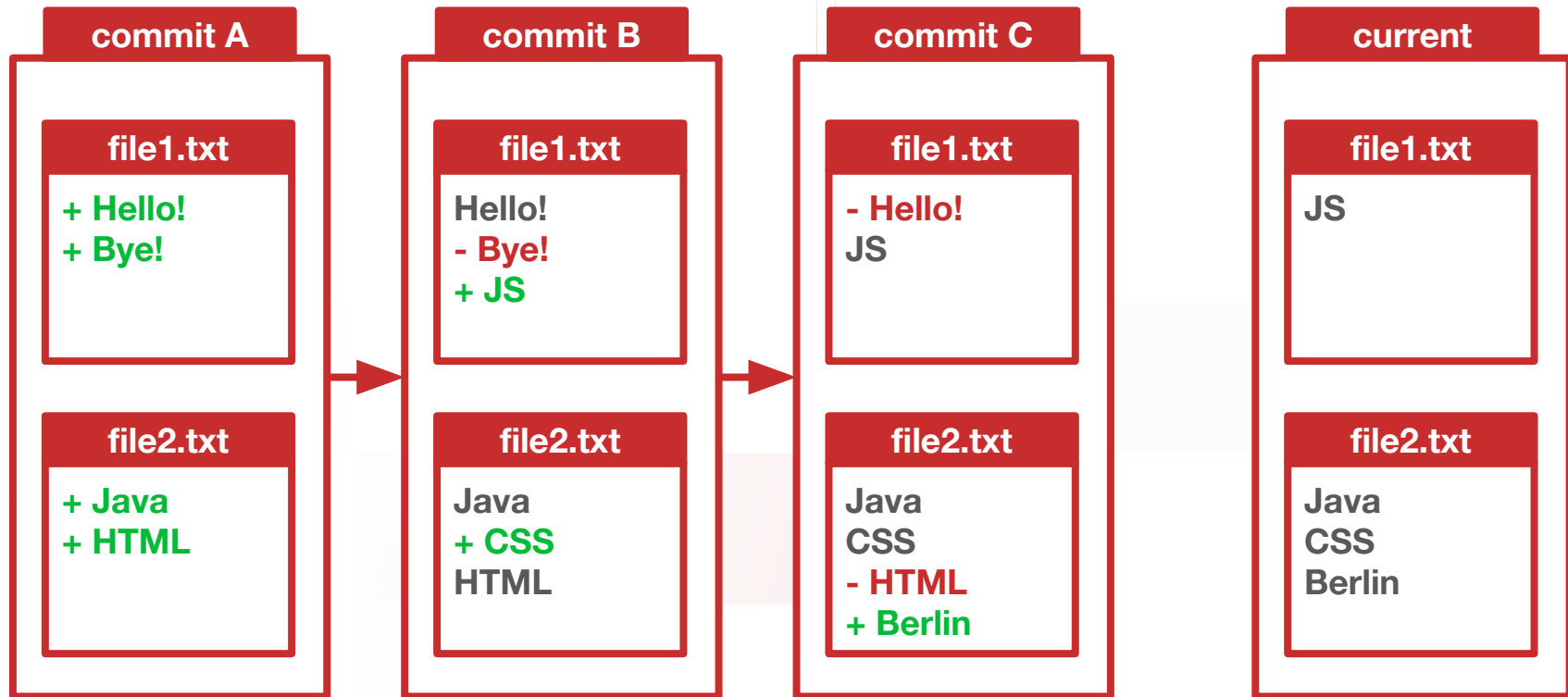
■ Добавление файла в индекс

■ Создание коммита

■ Отправка изменений

■ Файл .gitignore

# КОММИТЫ



# СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Коммиты и их значение

# ПОИГРАЕМ ;)

Система контроля версий

Репозиторий


Linux-команды

Коммиты

Команды git

# ДОМАШНЕЕ ЗАДАНИЕ





# **Ваша новая IT-профессия – Ваш новый уровень жизни**

Программирование с нуля в  
немецкой школе AIT TR GmbH