# Лабораторная работа №9: Практическая работа с Git

**Цель работы**: Познакомить студентов с основными операциями работы с удаленными репозиториями в Git, включая клонирование, добавление удаленных репозиториев, отправку изменений и работу с ветками на удаленном репозитории.

## Задание 1: Клонировать удаленный репозиторий::

## 

## 

1. Найти удаленный репозиторий на GitHub, GitLab или другой платформе.
2. На локальном компьютере выполнить команду git clone <URL удаленного репозитория>.
3. Убедиться, что репозиторий успешно склонирован на локальный компьютер.

## Задание 2: Добавить удаленный репозиторий:

## 

1. В командной строке выполнить команду git remote add origin <URL удаленного репозитория> для добавления удаленного репозитория в локальный.

## Задание 3: Отправить изменения на удаленный репозиторий:

## 

1. Выполнить команду git push origin <branch\_name> для отправки изменений из локального репозитория на удаленный.
2. Проверить, что изменения успешно отправлены на удаленный репозиторий.

## Задание 4: Работа с ветками на удаленном репозитории:

1. Создать новую ветку локально с помощью команды git checkout -b <branch\_name>.
2. Запушить новую ветку на удаленный репозиторий с помощью команды git push origin <branch\_name>.
3. Удалить ветку из удаленного репозитория с помощью команды git push origin --delete <branch\_name>.

**Задание 5: Получение изменений из удаленного репозитория::**

1. Выполнить команду git pull origin <branch\_name> для получения изменений из удаленного репозитория на локальный.
2. Убедиться, что изменения успешно применены к локальному репозиторию.

# Вопросы для самопроверки

1. Что такое система контроля версий и для чего она используется?

Система контроля версий (СКВ) — **это инструмент, который сохраняет историю изменений в файлах проекта**, позволяя разработчикам отслеживать изменения и управлять версиями кода. [1](https://simpleone.ru/glossary/version-control-system)

**Использование СКВ позволяет решить следующие задачи**:

* **Отслеживание изменений**. Каждое изменение в проекте фиксируется в виде коммита, что позволяет легко отслеживать, какие именно изменения были внесены, кем и когда. [2](https://sky.pro/wiki/javascript/sistemy-kontrolya-versij-zachem-oni-nuzhny-i-kak-ih-ispolzovat/)
* **Восстановление предыдущих версий**. Система контроля версий сохраняет историю изменений, что позволяет разработчикам легко возвращаться к предыдущим версиям кода. [3](https://itanddigital.ru/bloghrconsulting/tpost/836hhhzyu1-chto-takoe-sistema-kontrolya-versii)
* **Слияние изменений**. Когда несколько разработчиков работают над одним проектом, СКВ позволяет сливать их изменения в одну общую версию. [3](https://itanddigital.ru/bloghrconsulting/tpost/836hhhzyu1-chto-takoe-sistema-kontrolya-versii)
* **Контроль доступа**. СКВ позволяет определить, кто имеет доступ к проекту и какие права у разных пользователей. Это обеспечивает безопасность и конфиденциальность проекта. [3](https://itanddigital.ru/bloghrconsulting/tpost/836hhhzyu1-chto-takoe-sistema-kontrolya-versii)
* **Лёгкость восстановления после сбоев**. Если произошла ошибка или сбой в системе, СКВ может быть использована для восстановления проекта до последней стабильной версии. [3](https://itanddigital.ru/bloghrconsulting/tpost/836hhhzyu1-chto-takoe-sistema-kontrolya-versii)

Использование СКВ помогает улучшить эффективность работы, обеспечить целостность кода и сделать разработку программного обеспечения более организованной и управляемой

2. Какие основные преимущества использования Git перед другими системами контроля версий?

Основные преимущества использования Git перед другими системами контроля версий включают:

* **Независимость и гибкость**. Каждый разработчик имеет локальную копию репозитория, что позволяет работать без постоянного подключения к сети. [3](https://telegra.ph/Kakie-preimushchestva-ispolzovaniya-git-pered-drugimi-sistemami-kontrolya-versij-Git-Pochemu-ehto-korol-kontrolya-versij--i-ego--01-01)
* **Параллельная работа**. Git позволяет создавать ветки, что даёт возможность параллельно разрабатывать новые функции или исправлять ошибки, не затрагивая основную кодовую базу. [3](https://telegra.ph/Kakie-preimushchestva-ispolzovaniya-git-pered-drugimi-sistemami-kontrolya-versij-Git-Pochemu-ehto-korol-kontrolya-versij--i-ego--01-01)
* **Безопасность и надёжность**. Git хранит полную историю изменений, что позволяет легко откатиться к любой предыдущей версии. Это защищает проект от ошибок и позволяет восстановить код в случае проблем. [3](https://telegra.ph/Kakie-preimushchestva-ispolzovaniya-git-pered-drugimi-sistemami-kontrolya-versij-Git-Pochemu-ehto-korol-kontrolya-versij--i-ego--01-01)
* **Эффективное сотрудничество**. Git упрощает процесс слияния изменений от разных разработчиков, а также предоставляет инструменты для разрешения конфликтов, если они возникают. [3](https://telegra.ph/Kakie-preimushchestva-ispolzovaniya-git-pered-drugimi-sistemami-kontrolya-versij-Git-Pochemu-ehto-korol-kontrolya-versij--i-ego--01-01)
* **Скорость**. Git работает очень быстро, особенно при работе с большими проектами. Это связано с тем, что он сохраняет данные в виде снимков состояния, а не в виде списка изменений. [3](https://telegra.ph/Kakie-preimushchestva-ispolzovaniya-git-pered-drugimi-sistemami-kontrolya-versij-Git-Pochemu-ehto-korol-kontrolya-versij--i-ego--01-01)
* **Открытый исходный код**. Git является проектом с открытым исходным кодом, что означает его бесплатное использование и постоянное развитие благодаря вкладу сообщества.

3. Что такое команда "git clone" и как она используется?

Команда **git clone** в Git **используется для создания копии существующего репозитория**.  [1](https://sky.pro/wiki/profession/git-osnovnye-komandy-i-principy-raboty/)

Она позволяет скачать все файлы и историю изменений из удалённого репозитория на локальный компьютер.  [1](https://sky.pro/wiki/profession/git-osnovnye-komandy-i-principy-raboty/)

**Пример использования**:

git clone https://github.com/user/repository.git

. Эта команда создаст директорию

repository

 и скопирует в неё все файлы и историю изменений из удалённого репозитория.  [1](https://sky.pro/wiki/profession/git-osnovnye-komandy-i-principy-raboty/)

После клонирования разработчик получает **полноценный репозиторий Git** с собственной историей и файлами, полностью обособленный от исходного репозитория.

4. Какая команда используется для добавления файлов в индекс в Git?

Git add

5. Как создать новую ветку в Git и что означает "ветвление" в контексте системы контроля версий?

**Чтобы создать новую ветку в Git, нужно использовать команду git branch**. Например, для создания ветки с именем testing нужно выполнить команду

git branch testing

. В результате создаётся новый указатель на текущий коммит. [1](https://git-scm.com/book/ru/v2/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%9E-%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B8-%D0%B2-%D0%B4%D0%B2%D1%83%D1%85-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%85)[4](https://www.atlassian.com/ru/git/tutorials/using-branches)

**Ветвление** в контексте системы контроля версий означает **возможность работать над разными версиями проекта**. Используя ветвление, можно отклониться от основной линии разработки и продолжать работу независимо от неё, не вмешиваясь в основную линию. [3](https://otus.ru/nest/post/770/)[1](https://git-scm.com/book/ru/v2/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%9E-%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B8-%D0%B2-%D0%B4%D0%B2%D1%83%D1%85-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%85)

**В Git ветвление легковесное**: операция создания ветки выполняется почти мгновенно, переключение между ветками также происходит быстро. Вместо того чтобы копировать файлы из каталога в каталог, Git хранит ветку в виде ссылки на коммит.

6. Что означает команда "git push" и как она используется для отправки изменений на удаленный репозиторий?

Команда **git push** в системе Git **позволяет отправлять локальную ветку на удалённый репозиторий**.  [1](https://selectel.ru/blog/tutorials/what-is-git-push-and-how-to-use-it/" \t "_blank)

Она помогает разработчикам синхронизироваться в команде, а именно отправляет проделанные изменения.  [1](https://selectel.ru/blog/tutorials/what-is-git-push-and-how-to-use-it/)

**Некоторые варианты использования команды git push**:

* **Отправка изменений в другую ветку**. Их названия последовательно указываются через двоеточие. Например,

git push origin master:remote\_branch

, где

master

 — название локальной ветки, а

remote\_branch

 — аналогичная ветка удалённого репозитория, но с другим именем. [3](https://timeweb.cloud/tutorials/git/git-push-kak-rabotaet-komanda)

* **Отправка всех веток сразу**. Для этого вместо имени ветки указывается флаг

--all

. Например,

git push origin --all

. Git отправит свежие коммиты в соответствующие ветки удалённого репозитория — те, что имеют идентичные имена. [3](https://timeweb.cloud/tutorials/git/git-push-kak-rabotaet-komanda)

* **Отправка текущей ветки**. Можно использовать краткую форму команды, которая отправляет в удалённый репозиторий текущую (т. е. активную) ветку. Например,

git push origin HEAD

7. Как можно получить изменения из удаленного репозитория на локальный с помощью Git?

**Чтобы получить изменения из удалённого репозитория на локальный с помощью Git, можно использовать команду git fetch**. Она извлекает информацию о последних изменениях, внесённых в удалённый репозиторий, и обновляет ветки удалённого отслеживания. [1](https://smartiqa.ru/courses/git/lesson-6)[2](https://docs.github.com/ru/get-started/using-git/getting-changes-from-a-remote-repository)[3](https://selectel.ru/blog/tutorials/git-fetch-command-how-is-it-different-from-git-pull/)

**Пример использования**:

git fetch origin

. Команда не обновляет рабочую копию в соответствии с удалённым репозиторием, а скачивает все необходимые файлы в директорию .git/objects. [1](https://smartiqa.ru/courses/git/lesson-6)

**Также для получения изменений и одновременного обновления рабочей копии можно использовать команду git pull**. Она запускает команду git fetch для загрузки содержимого из указанного удалённого репозитория, а затем выполняет команду git merge, объединяющую ссылки и указатели удалённого содержимого в новый локальный коммит слияния.

8. Что такое merge (слияние) в Git и какое его предназначение?

**Merge (слияние) в Git** — это **способ объединения изменений из одной ветки в другую**. При выполнении слияния Git создаёт новый коммит, который объединяет изменения из исходной ветки в целевую. Этот коммит явно указывает на то, что произошло объединение веток. [1](https://proglib.io/p/komandy-git-merge-i-git-rebase-preimushchestva-nedostatki-i-klyuchevye-razlichiya-2024-11-01)

**Предназначение merge**:

* **Сохранение истории изменений**. Команда позволяет отслеживать точную историю всех изменений и решений. [1](https://proglib.io/p/komandy-git-merge-i-git-rebase-preimushchestva-nedostatki-i-klyuchevye-razlichiya-2024-11-01)
* **Подходит для совместных проектов**, где важно понимать контекст изменений. [1](https://proglib.io/p/komandy-git-merge-i-git-rebase-preimushchestva-nedostatki-i-klyuchevye-razlichiya-2024-11-01)
* **Используется в проектах, где важно отслеживать общедоступные и общие ветки**. Коммиты слияния обеспечивают чёткую историю для таких веток. [1](https://proglib.io/p/komandy-git-merge-i-git-rebase-preimushchestva-nedostatki-i-klyuchevye-razlichiya-2024-11-01)

Слияние не затрагивает сливаемую ветку, то есть она остаётся в том же состоянии. Это позволяет потом продолжить работу с ней.

9. Какое предназначение у команды "git pull"?

Команда

git pull

 в Git **предназначена для извлечения изменений из удалённого репозитория и автоматического объединения их с текущей веткой**.  [4](https://itproger.com/spravka/git/pull)

Это упрощает процесс обновления локального репозитория с изменениями из удалённого, что **особенно полезно в ситуациях, когда работают в команде и хотят быть в курсе последних обновлений**.

10. Какая команда используется для удаления ветки из удаленного репозитория в Git?

Для удаления ветки из удалённого репозитория в Git используется команда **git push <удаленное-имя-репозитория> --delete <имя-ветки>**.





