# Настройка репозитория

[git init](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository) [git clone](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository) [git config](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository)

Этот обучающий материал включает в себя обзор настройки репозитория в системе контроля версий Git. На этой странице вы узнаете, как инициализировать репозиторий Git для нового или существующего проекта. Ниже представлены примеры жизненного цикла для репозиториев, созданных локально и клонированных из удаленных репозиториев. Для работы с этим руководством требуются начальные знания о работе с интерфейсом командной строки.

В данном руководстве обсуждаются следующие основные вопросы:

* Инициализация нового репозитория Git
* Клонирование существующего репозитория Git
* Коммит измененной версии файла в репозиторий
* Конфигурирование репозитория Git для удаленной совместной работы
* Распространенные команды для управления версиями Git

По окончании данного модуля вы должны уметь создавать репозиторий Git, использовать основные команды Git, выполнять коммит измененного файла, просматривать историю проекта и настраивать соединение с сервисом хостинга Git (Bitbucket).

**Что такое репозиторий Git?**

[Репозиторий Git](https://bitbucket.org/product/ru/code-repository) — это виртуальное хранилище проекта. В нем можно хранить версии кода для доступа по мере необходимости.

**Инициализация нового репозитория: git init**

Для создания нового репозитория используется команда git init. Команду git init выполняют только один раз для первоначальной настройки нового репозитория. Выполнение команды приведет к созданию нового подкаталога .git в вашем рабочем каталоге. Кроме того, будет создана новая главная ветка.

**Создание версии существующего проекта с использованием нового репозитория Git**

В этом примере предполагается, что у вас уже есть папка проекта, в которой вы и хотите создать репозиторий. Выполните команду cd для перехода к папке проекта, а затем выполните команду git init.

cd /path/to/your/existing/code   
git init

Указание в команде git init существующего каталога проекта приведет к исполнению описанной выше инициализации, но только на уровне этого каталога проекта.

git init <project directory>

Перейдите на страницу [git init](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-init), чтобы получить подробные сведения о команде git init.

**Клонирование существующего репозитория: git clone**

Если проект уже настроен в центральном репозитории, наиболее распространенным способом создать его локальный клон является команда clone. Клонирование, как и команда git init, обычно выполняется один раз. Получив рабочую копию, разработчик в дальнейшем выполняет все операции [контроля версий](https://bitbucket.org/product/ru/version-control-software) из своего локального репозитория.

git clone <repo url>

Команду git clone выполняют для создания копии (клонирования) удаленного репозитория. В качестве параметра в команду git clone передается URL-адрес репозитория. Git поддерживает несколько различных сетевых протоколов и соответствующих форматов URL-адресов. В этом примере используется SSH-протокол Git. URL-адреса SSH в Git имеют следующий шаблон: git@HOSTNAME:USERNAME/REPONAME.git

Пример URL-адреса SSH в Git имеет вид: git@bitbucket.org:rhyolight/javascript-data-store.git, а ниже приведены значения шаблонных параметров:

* HOSTNAME: bitbucket.org
* USERNAME: rhyolight
* REPONAME: javascript-data-store

После исполнения команды последние версии файлов из главной ветки удаленного репозитория будут загружены и помещены в новый каталог. Имя нового каталога будет соответствовать параметру REPONAME. В данном случае это javascript-data-store. В каталоге будет вся история удаленного репозитория и только что созданная главная ветка.

Дополнительную информацию об использовании команды git clone и поддерживаемых форматах URL-адресов в Git см. на [странице git clone](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-clone).

**Сохранение изменений в репозитории: git add и git commit**

У вас появился репозиторий, созданный путем клонирования или инициализации. Теперь вы можете выполнять коммиты изменений в версиях файлов. В следующем примере предполагается, что вы настроили проект в каталоге /path/to/project. В этом примере предлагаются следующие шаги.

* Измените каталоги на /path/to/project
* Создайте новый файл CommitTest.txt с текстом ~«тест для обучения работе с Git»~
* С помощью команды git add добавьте файл CommitTest.txt в репозиторий проиндексированных файлов
* Создайте новый коммит с комментарием, описывающим, что именно было изменено в коммите

cd /path/to/project   
echo "test content for git tutorial" >> CommitTest.txt   
git add CommitTest.txt   
git commit -m "added CommitTest.txt to the repo"

По завершении этого примера файл CommitTest.txt добавится к истории репозитория, и репозиторий будет отслеживать последующие изменения в файле.

В этом примере представлены две новые команды в Git: add и commit. Этот очень упрощенный пример. Подробнее обе команды объяснены на страницах [git add](https://www.atlassian.com/ru/git/tutorials/saving-changes) и [git commit](https://www.atlassian.com/ru/git/tutorials/saving-changes/git-commit). Команду git add часто используют с флагом --all. Команда git add --all добавляет все измененные и неотслеживаемые файлы в репозиторий и обновляет дерево изменений репозитория.

**Совместная работа в разных репозиториях: git push**

Важно понимать, что рабочая копия в Git существенно отличается от рабочей копии, получаемой при загрузке исходного кода из репозитория SVN. В отличие от SVN, в Git нет разницы между рабочими копиями и центральным репозиторием — все они являются полноценными [репозиториями Git](https://bitbucket.org/product/ru/code-repository).

Поэтому совместная работа в Git принципиально отличается от совместной работы в SVN. В SVN работа строится на отношении между центральным репозиторием и рабочей копией, а модель совместной работы в Git основана на взаимодействии между репозиториями. Вместо загрузки рабочей копии в центральный репозиторий SVN в Git вы отправляете коммиты из одного репозитория в другой или копируете их в обратном направлении.

Вы легко можете задавать особую роль определенным репозиториям Git. Например, обозначив один из репозиториев Git как «центральный», вы можете воспроизвести централизованный процесс с использованием Git. Такой подход требует общих договоренностей, он не встроен в саму систему контроля версий.

**Сравнение чистых и клонированных репозиториев**

Если в предыдущем разделе («Инициализация нового репозитория») для настройки локального репозитория вы использовали команду git clone, ваш репозиторий уже готов к удаленной совместной работе. Команда git clone автоматически настроит репозиторий, в котором значение remote будет соответствовать URL-адресу Git, из которого был клонирован репозиторий. Это означает, что после изменений файла и выполнения коммита вы можете сразу выполнить команду git push, чтобы отправить эти изменения в удаленный репозиторий.

Если вы использовали команду git init для создания репозитория с нуля, у вас не будет удаленного репозитория, в который можно помещать изменения. Зачастую для инициализации нового репозитория пользователь переходит на сервис Git-хостинга (например, Bitbucket) и создает репозиторий там. Данный сервис предоставит URL-адрес Git, который затем можно добавить в локальный репозиторий Git. После этого можно выполнять команду git push в репозиторий на хостинге. После создания удаленного репозитория на выбранном хостинге вам понадобится обновить локальный репозиторий, выполнив привязку. Этот процесс описывается далее в руководстве по установке и настройке.

Если вы предпочитаете поддерживать собственный удаленный репозиторий, вам нужно создать «чистый репозиторий». Для этого команды git init и git clone принимают аргумент --bare. Наиболее популярная причина использования чистого репозитория — создание удаленного центрального репозитория Git

**Конфигурирование и настройка: git config**

После настройки удаленного репозитория его URL-адрес нужно добавить в локальный файл git config, а также создать вышестоящую ветку для локальных веток. Такую возможность предоставляет команда git remote.

git remote add <remote\_name> <remote\_repo\_url>

Эта команда привяжет удаленный репозиторий по адресу к ссылке в вашем локальном репозитории . После привязки удаленного репозитория в него можно будет отправлять локальные ветки с помощью команды push.

git push -u <remote\_name> <local\_branch\_name>

Эта команда поместит ветку локального репозитория с именем < local\_branc\_name > в удаленный репозиторий < remote\_name >.

Помимо конфигурирования URL-адреса удаленного репозитория, вам может потребоваться установить глобальные параметры Git, например имя пользователя или электронный адрес. Команда git config позволяет настроить инсталляцию Git (или отдельный репозиторий) из командной строки. С помощью этой команды можно установить любые настройки: от информации о пользователе до его предпочтений и характеристик репозитория. Ниже перечислены распространенные варианты конфигурации.

Git хранит варианты конфигурации в трех различных файлах, позволяющих ограничивать область видимости на уровне отдельных репозиториев (локальный), пользователя (глобальный) или всей системы (системный):

* Локальный: /.git/config — настройки на уровне репозитория.
* Глобальный: /.gitconfig — настройки на уровне пользователя. Здесь хранятся настройки с флагом --global.
* Системный: $(prefix)/etc/gitconfig — настройки на уровне всей системы.

Укажите имя автора, которое будет использоваться для всех коммитов в текущем репозитории. Обычно для настройки параметров конфигурации для текущего пользователя используется флаг --global.

git config --global user.name <name>

Эта команда задает имя автора, которое будет использоваться для всех коммитов, выполненных текущим пользователем.

Добавление аргумента --local или выполнение команды без параметра уровня конфигурации приведет к установке значения user.name для текущего локального репозитория.

git config --local user.email <email>

Эта команда задает адрес электронной почты автора, который будет использоваться для всех коммитов, выполненных текущим пользователем.

git config --global alias.<alias-name> <git-command>

Создайте быстрые клавиши для команды Git. Это мощная возможность для создания собственных комбинаций клавиш для часто используемых команд Git. Ниже показан упрощенный пример:

git config --global alias.ci commit

Так создается команда ci, которую можно использовать как сокращение команды git commit. Подробнее об алиасах в Git см. на [странице git config](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-config).

git config --system core.editor <editor>

Выберите текстовый редактор, используемый для таких команд, как git commit, для всех пользователей текущего компьютера. Аргумент должен представлять собой команду, запускающую нужный редактор (например, vi). В этом примере представлен аргумент --system. Аргумент --system устанавливает настройку на уровне всей системы, включая всех пользователей и все репозитории на компьютере. Дополнительную информацию об уровнях конфигурации см. на [странице удаленной работы с git](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-config).

git config --global --edit

В текстовом редакторе откройте файл глобальной конфигурации для редактирования вручную. Подробное руководство по настройке текстового редактора для Git см. на [странице Git config](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-config).

### Пояснения

Все варианты конфигурации сохраняются в обычных текстовых файлах, так что команда git config — это всего лишь удобный интерфейс командной строки. Как правило, установку Git следует настраивать только при начале работы на новом компьютере. В подавляющем большинстве случаев понадобится только флаг --global. Одно из важных исключений — необходимость переписать электронный адрес автора. Вы можете поставить личный электронный адрес для личных репозиториев и репозиториев с открытым исходным кодом, а рабочий электронный адрес — для рабочих репозиториев.

Git хранит варианты конфигурации в трех различных файлах, что позволяет ограничивать область видимости на уровне отдельных репозиториев, пользователей или всей системы.

* /.git/config — настройки на уровне репозитория.
* ~/.gitconfig — личные настройки пользователя. Здесь хранятся настройки с флагом --global.
* $(prefix)/etc/gitconfig — настройки на уровне всей системы.

Если параметры, указанные в этих файлах, конфликтуют, локальные настройки переопределяют пользовательские настройки, которые в свою очередь переопределяют системные настройки. Если вы откроете один из этих файлов, вы увидите нечто подобное:

[user] name = John Smith email = john@example.com [alias] st = status co = checkout br = branch up = rebase ci = commit [core] editor = vim

Вы можете изменить эти значения вручную, эффект будет аналогичен использованию команды git config.

### Пример

В первую очередь после установки Git требуется указать свое имя и адрес электронной почты, а также настроить некоторые параметры по умолчанию. Пример типичной начальной конфигурации показан далее.

Представьтесь репозиторию Git с помощью команды git config

git --global user.name "John Smith" git config --global user.email john@example.com

Выберите любимый текстовый редактор

git config --global core.editor vim

Добавьте алиасы по типу SVN

git config --global alias.st status   
git config --global alias.co checkout   
git config --global alias.br branch   
git config --global alias.up rebase   
git config --global alias.ci commit

Создастся файл ~ /.gitconfig, описанный в предыдущем разделе. Подробную информацию о команде git config см. на [странице Git config](https://www.atlassian.com/ru/git/tutorials/setting-up-a-repository/git-config).