Шпаргалка по Git (от базовых до продвинутых концепций)

Шпаргалка по Git (от базовых до продвинутых концепций)

Команды установки Git

Вот команды установки Git для различных операционных систем:

Команды	**Описание**
Автономный установщик Git для Windows.	Для более подробной информации**Читайте здесь**
$ brew install git	Установка Git с помощью**Homebrew**на Mac OS
$ sudo port selfupdate	Установка Git с помощью**MacPorts**на Mac OS
$ sudo apt-get install git	Команда установки для Linux
$ git –version	Показывает текущую версию вашего Git

Конфигурация и настройка Git

Вот команды конфигурации и настройки Git:

Команды	**Описание**
git config –global user.name "Ваше Имя"	Установка вашего имени пользователя глобально.
git config –global user.email "youremail@example.com"	Установка вашей электронной почты глобально.
git config –global color.ui auto –	Установка для отображения цветного вывода в терминале
git help	Отображение основной справочной документации, показывающей список часто используемых команд Git.

Инициализация репозитория

Вот команды инициализации репозитория Git:

Команды	Описание
git init	Инициализирует новый репозиторий Git в текущем каталоге.
git init <directory>	Создает новый репозиторий Git в указанном каталоге.
git clone <repository_url>	Клонирует репозиторий с удаленного сервера на вашу локальную машину.
git clone –branch <branch_name> <repository_url>	Клонирует определенную ветку из репозитория.

Основные команды Git

Вот некоторые основные команды Git:

Команды	Описание
git add <file>	Добавляет конкретный файл в область подготовки.
git add . или git add –all	Добавляет все измененные и новые файлы в область подготовки.
git status	Показывает текущее состояние вашего репозитория, включая отслеживаемые и неотслеживаемые файлы, измененные файлы и информацию о ветке.
git status –ignored	Отображает игнорируемые файлы в дополнение к обычному выводу состояния.
git diff	Показывает изменения между рабочим каталогом и областью подготовки (индексом).
git diff <commit1> <commit2>	Отображает различия между двумя коммитами.
git diff –staged или git diff –cached	Отображает изменения между областью подготовки (индексом) и последним коммитом.
git diff HEAD	Отображает разницу между текущим каталогом и последним коммитом

git commit	Создает новый коммит с изменениями в области подготовки и открывает текстовый редактор по умолчанию для добавления сообщения коммита.
git commit -m "<message>" или git commit –message "<message>"	Создает новый коммит с изменениями в области подготовки и указывает сообщение коммита в строке.
git commit -a или git commit –all	Коммитит все измененные и удаленные файлы в репозитории без явного использования git add для подготовки изменений.
git notes add	Создает новую заметку и связывает ее с объектом (коммит, тег и т. д.).
git restore <file>	Восстанавливает файл в рабочем каталоге до его состояния в последнем коммите.
git reset <commit>	Перемещает указатель ветки на указанный коммит, сбрасывая область подготовки и рабочий каталог, чтобы они соответствовали указанному коммиту.
git reset –soft <commit>	Перемещает указатель ветки на указанный коммит, сохраняя изменения в области подготовки и рабочем каталоге.
git reset –hard <commit>	Перемещает указатель ветки на указанный коммит, отбрасывая все изменения в области подготовки и рабочем каталоге, фактически сбрасывая репозиторий до указанного коммита.
git rm <file>	Удаляет файл как из рабочего каталога, так и из репозитория, подготавливая удаление.
git mv	Перемещает или переименовывает файл или каталог в вашем репозитории Git.

> Также проверьте: Основные команды Git с примерами

Коммит Git (обновленные команды)

Вот некоторые обновленные команды для коммита Git:

Команды	**Описание**

git commit -m "feat: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения коммита новой функции в репозитории.
git commit -m "fix: message"	Создать новый коммит в репозитории Git с конкретным сообщением для исправления ошибок в кодовой базе
git commit -m "chore: message"	Создать новый коммит в репозитории Git с конкретным сообщением для отображения рутинных задач или обслуживания.
git commit -m "refactor: message"	Создать новый коммит в репозитории Git с конкретным сообщением для изменения кодовой базы и улучшения структуры.
git commit -m "docs: message"	Создать новый коммит в репозитории Git с конкретным сообщением для изменения документации.
git commit -m "style: message"	Создать новый коммит в репозитории Git с конкретным сообщением для изменения стиля и форматирования кодовой базы.
git commit -m "test: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения изменений, связанных с тестированием.
git commit -m "perf: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения изменений, связанных с производительностью.
git commit -m "ci: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения изменений, связанных с системой непрерывной интеграции (CI).
git commit -m "build: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения изменений, связанных с процессом сборки.
git commit -m "revert: message"	Создать новый коммит в репозитории Git с конкретным сообщением для обозначения изменений, связанных с отменой предыдущего коммита.

Ветвление и слияние

Вот некоторые команды ветвления и слияния Git:

Команды	**Описание**

git branch	Перечисляет все ветки в репозитории.
git branch <branch-name>	Создает новую ветку с указанным именем.
git branch -d <branch-name>	Удаляет указанную ветку.
git branch -a	Перечисляет все локальные и удаленные ветки.
git branch -r	Перечисляет все удаленные ветки.
git checkout <branch-name>	Переключается на указанную ветку.
git checkout -b <new-branch-name>	Создает новую ветку и переключается на нее.
git checkout — <file>	Отменяет изменения, внесенные в указанный файл, и возвращает его к версии в последнем коммите.
git merge <branch>	Объединяет указанную ветку с текущей веткой.
git log	Отображает историю коммитов текущей ветки.
git log <branch-d	Отображает историю коммитов указанной ветки.
git log –follow <file>	Отображает историю коммитов файла, включая его переименования.
git log –all	Отображает историю коммитов всех веток.
git stash	Сохраняет изменения в рабочем каталоге, позволяя переключиться на другую ветку или коммит без фиксации изменений.
git stash list	Перечисляет все сохраненные изменения в репозитории.
git stash pop	Применяет и удаляет последнее сохраненное изменение из списка stash.
git stash drop	Удаляет последнее сохраненное изменение из списка stash.
git tag	Перечисляет все теги в репозитории.
git tag <tag-name>	Создает легковесный тег на текущем коммите.
git tag <tag-name> <commit>	Создает легковесный тег на указанном коммите.
git tag -a <tag-name> -m "<message>"	Создает аннотированный тег на текущем коммите с пользовательским сообщением.

Удаленные репозитории

Вот некоторые команды удаленных репозиториев Git:

Команды	Описание
git fetch	Извлекает изменения из удаленного репозитория, включая новые ветки и коммиты.
git fetch <remote>	Извлекает изменения из указанного удаленного репозитория.
git fetch –prune	Удаляет все удаленные ветки, которые больше не существуют в удаленном репозитории.
git pull	Извлекает изменения из удаленного репозитория и объединяет их с текущей веткой.
git pull <remote>	Извлекает изменения из указанного удаленного репозитория и объединяет их с текущей веткой.
git pull –rebase	Извлекает изменения из удаленного репозитория и выполняет перебазирование текущей ветки на обновленную ветку.
git push	Отправляет локальные коммиты в удаленный репозиторий.
git push <remote>	Отправляет локальные коммиты в указанный удаленный репозиторий.
git push <remote> <branch>	Отправляет локальные коммиты в указанную ветку удаленного репозитория.
git push –all	Отправляет все ветки в удаленный репозиторий.
git remote	Перечисляет все удаленные репозитории.
git remote add <name> <url>	Добавляет новый удаленный репозиторий с указанным именем и URL.

Сравнение в Git

Вот некоторые команды сравнения Git:

Команды	Описание
git show	Показывает детали конкретного коммита, включая его изменения.
git show <commit>	Показывает детали указанного коммита, включая его изменения.

Управление историей Git

Вот некоторые команды управления историей Git:

Команды	**Описание**
git revert <commit>	Создает новый коммит, который отменяет изменения, внесенные указанным коммитом.
git revert –no-commit <commit>	Отменяет изменения, внесенные указанным коммитом, но не создает новый коммит.
git rebase <branch>	Повторно применяет коммиты текущей ветки на конец указанной ветки.