

# Лабораторная работа № 2. Управление версиями git

---

фio:ван яо

группа:НПМбд-02-21

билет:1032215430

## Содержание

---

- [Введение](#)
- [Часто используемые команды](#)
- [Домашнее задание](#)
- [Контрольные вопросы](#)
- [Выводы](#)

## Введение

---

Овладение навыками использования Git очень важно, особенно в процессе разработки программного обеспечения и совместной работы.

## Часто используемые команды

---

### 1. Установка Git

Сначала убедитесь, что Git установлен на вашем компьютере. Если Git еще не установлен, скачайте и установите его с официального сайта:

[Официальный сайт Git.](#)

### 2. Инициализация Git-репозитория

Перед началом использования Git необходимо настроить ваше имя пользователя и электронную почту:

```
git config --global user.name "Ваше имя пользователя"
```

```
git config --global user.email "Ваш адрес электронной почты"
```

### 3. Инициализация Git-репозитория

Инициализируйте новый Git-репозиторий в каталоге проекта:

```
cd /path/to/your/project
```

```
git init
```

## 4. Добавление файлов

Добавьте файлы в Git-репозиторий:

```
git add <file>
```

## 5. Фиксация изменений

Зафиксируйте добавленные файлы в Git-репозиторий:

```
git commit -m "Сообщение о фиксации"
```

## 6. Просмотр состояния

Просмотрите текущее состояние репозитория:

```
git status
```

## 7. Просмотр истории фиксаций

Просмотрите историю фиксаций:

```
git log
```

###8. Создание ветки

Создайте новую ветку:

```
git branch new-branch
```

###9. Объединение веток

Объедините ветку с главной веткой:

```
git checkout main
```

```
git merge new-branch
```

## 10. Удаление ветки

Удалите ветку

```
git branch -d new-branch
```

## 11. Клонирование удаленного репозитория

Клонируйте удаленный репозиторий на локальную машину:

```
git clone https://github.com/yourusername/yourrepository.git
```

###12. Отправка изменений на удаленный репозиторий

Отправьте локальные изменения на удаленный репозиторий:

```
git push origin main
```

## 13. Получение изменений из удаленного репозитория

Получите последние изменения из удаленного репозитория:

```
git pull origin main
```

## 14. Разрешение конфликтов

Найдите файлы с конфликтами, вручную исправьте их и снова зафиксируйте::

```
git add <conflicted-file>
```

```
git commit -m "Разрешение конфликтов"
```

## 15. Возврат к предыдущей версии

Вернитесь к предыдущей версии:

```
git checkout <commit-hash>
```

## 16. Управление тегами

Отправьте тег на удаленный репозиторий:

```
git push origin v1.0
```

###17. Отмена изменений

Отмените нефиксированные изменения:

```
git checkout -- <file>
```

## 18. Очистка рабочего каталога

Очистите рабочий каталог от неотслеживаемых файлов:

```
git clean -f
```

## 19. Проверка настроек Git

Проверьте настройки Git:

```
git config --list
```

## 20. Экспорт и импорт Git-репозитория

Экспортируйте историю Git-репозитория:

```
git bundle create my-bundle-name.bundle main
```

# Домашнее задание

<input type="checkbox"/> 名称	修改日期	类型	大小
 new	2024/9/3 2:23	文件夹	
 id_ed25519	2024/9/3 2:29	文件	1 KB
<input checked="" type="checkbox"/>  id_ed25519.pub	2024/9/3 2:29	PUB 文件	1 KB
 id_rsa	2024/9/3 2:25	文件	4 KB
 id_rsa.pub	2024/9/3 2:25	PUB 文件	1 KB
 known_hosts	2024/9/3 2:30	文件	1 KB



```
MINGW32:/d/work/2024-2025/infosec
wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$ git push -u origin main
Counting objects: 8, done.
Delta compression using up to 18 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 700 bytes | 0 bytes/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To github.com:wangyao200036/infosec.git
 * [new branch]      main -> main
Branch main set up to track remote branch main from origin.

wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$ openssl genpkey -algorithm RSA -out pdp_private_key.pem
.....+++++
n.....+++++

wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$ openssl rsa -pubout -in pdp_private_key.pem -out pdp_public_key.pem
writing RSA key

wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$ cat pdp_private_key.pem
-----BEGIN PRIVATE KEY-----
MIICdWIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAKYmgRMFnAi+hM7A
WT68oMr290EqnPIOB5VnbCEQHRVj/R01zdHspFZ8RYAkboPYVmSJRS5A2VQ3rdk04
RwpAILyhpHXdsi5xcimykC8o2mQRbbwRvpkdkP9Gx3Y2OxqLdmp2zaozH4ruxQE+
5VLX9HQZjylKaPY38iWYFBpjfhTJAgMBAAECgYEAh91emQZyRLg+c67Yi7PFRJLt
PN024Tztuiv7R0Ic9jvwY+BhAPOIouw2DQY8QMjMRKP5cBTt01ckQxrtBJXR7fwm
ncRNAGhTk+/2ssxY2iVy2PqnZttWLkGa3mpgZlFMLhMkMYy4+6KlToPSy9hrPyN7
Z7Uqg6386w6hdQPlcs0CQqDb5Se1mTtc8+xNJTF8xNGfQLrQarTTtdoGyC5jtQSX
pjaqK0p16pPnl+DsMuzwTEFcai/bMabNLpgkBRXebw8vAKEAwW5MdHft4QTb06fZ
ZxGoS1DJaIs65Jg8s2+IiL2F/Krv4i+79xJlB9mf+eDJz8WShgBEgEOKNYJstoyI
5eddhWJAG8gf9Dmf3maSKh4kmbjZRNwhtr0dCsNmL3B3qcIJYDCd01w4ut40tJd
gHoOpXqgp2RjF3Z2dIwx9BsQYEH30QJAZ067cq1qG/X2WjKZQB0EH0WEbu/xiWnt
eAAAnkbB3MR0komsQHSMy0gB/GnveDSpeP8dAzqhD1pRT7sut0GwjXQJBAIjy4yn0
BqbUN4+GKUX1cj6oz+te5KWILAvfVU6R6/uDpwFRwiNAiJZb5CInzTHjquioXKq+
RkgOiZGb0e7XfIo=
-----END PRIVATE KEY-----

wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$ cat pdp_public_key.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCMJoETBZWivoTOWFk+vKDK9vdB
KpzyDgeVZ2whEB0VY/0dJc3R7KRwFEWAJGzj2FZkiUeQN1UN63ZNOEckQCC8oar1
3UoucXIpspHPKNpkEW28Eb6ZHZD/Rsd2Njsai3Zqds2qMx+K7sUBPuVS1/R0M48p
Smj2N/I1mBQaY34UyQIDAQAB
-----END PUBLIC KEY-----

wangy@MINGW32 /d/work/2024-2025/информационнаяБезопасность/infosec (main)
$
```

## Контрольные вопросы

### 1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

- Системы контроля версий (VCS, Version Control Systems) представляют собой программное обеспечение, которое используется для управления изменениями в файлах или наборах файлов на протяжении времени. Это позволяет пользователям сохранять историю изменений, откатываться к ранним версиям, работать над проектами совместно и управлять различными версиями проекта.
- Управление версиями файлов, Ветвление и слияние, Откат изменений

## 2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище (Repository) — это место, где система контроля версий хранит всю информацию о проекте, включая все версии файлов и историю изменений. Хранилище обычно содержит скрытую директорию (например, `.git` для Git), которая содержит все данные о версиях и метаданные.
- Commit (Фиксация) — это точка в истории проекта, которая представляет собой сохранённое состояние всех файлов в определённый момент времени. Когда вы делаете коммит, вы сохраняете текущее состояние файлов и добавляете сообщение, которое описывает внесённые изменения
- История (History) — это последовательность коммитов, которые были сделаны в проекте. История позволяет просматривать все изменения, внесённые в проект со временем.
- Рабочая копия (Working Directory) — это копия файлов проекта, с которой вы работаете прямо сейчас. Это место, где вы вносите изменения перед тем, как они будут добавлены в хранилище.

## 3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

- Централизованные системы контроля версий используют центральное хранилище (репозиторий), в котором хранится вся информация о проекте. Клиенты (разработчики) работают с локальными копиями этого хранилища, но все изменения должны быть отправлены в центральное хранилище для сохранения.
  - Простота
  - Управление доступом
- Децентрализованные системы контроля версий позволяют каждому разработчику иметь полную копию всего хранилища проекта. Это обеспечивает большую гибкость и надёжность.
  - Гибкость
  - Надёжность
- Примеры
  - Централизованные VCS (SVN)
  - Децентрализованные VCS (Git)

## 4. Опишите действия с VCS при единоличной работе с хранилищем.

- Инициализация нового репозитория
- Добавление и фиксация изменений
- Просмотр истории коммитов
- Откат к предыдущим версиям
- Создание и переключение веток
- Удаление коммитов и файлов

## 5. Опишите порядок работы с общим хранилищем VCS.

1. Клонирование удалённого репозитория
2. Проверка состояния репозитория
3. Создание и переключение веток
4. Внесение изменений и добавление файлов в индекс
5. Фиксация изменений (commit)
6. Синхронизация с удалённым репозиторием
7. Объединение веток (merge)
8. Разрешение конфликтов
9. Разрешение конфликтов
10. Проверка истории коммитов

## 6. Каковы основные задачи, решаемые инструментальным средством git?

1. Управление версиями файлов
2. Совместная работа
3. Контроль доступа
4. Безопасность и надёжность

## 7. Назовите и дайте краткую характеристику командам git.

1. git init
2. git clone
3. git add
4. git commit
5. git status

## 8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Примеры работы с локальным репозиторием

1. Инициализация нового репозитория
2. Добавление и фиксация изменений  
Примеры работы с локальным и удалённым репозиториями
3. Создание нового репозитория и отправка его на GitHub
4. Работа с ветками и слияние изменений



## 9. Что такое и зачем могут быть нужны ветви (branches)?

- Ветви в Git представляют собой указатели на конкретные коммиты. Каждая ветка указывает на последний коммит в этой ветке. Когда вы работаете в ветке, все ваши изменения и коммиты добавляются к этой ветке. Ветви позволяют создавать параллельные линии разработки, что упрощает управление проектом.
- 1. Разработка новых функций
- 2. Исправление ошибок

## 10. Как и зачем можно игнорировать некоторые файлы при commit?

1. Уменьшение размера репозитория
2. Защита конфиденциальной информации
3. Поддержание чистоты репозитория
4. Ускорение работы Git

## Выводы

---

Изучение команд Git полезно для разработки программного обеспечения и других работ, связанных с совместной работой и контролем версий. Git — это распределенная система контроля версий, которая позволяет отслеживать историю изменений файлов и поддерживает совместную работу нескольких человек.