Лабораторна робота № 1

Тема: Основи роботи з Git. Налаштування Git-GitHub середовища

Група: КН-24-1
Студент: Соломка Борис Олегович
Дата виконання: 09.11.2025

Мета роботи

Швидкий старт для роботи із системою контроля версій (СКВ) Git та віддаленим репозиторієм GitHub.

Завдання:

- Інсталювати Git, створювати локальний Git-репозиторій
- Створювати репозиторій на GitHub та зв'язувати його з локальним репозиторієм
- Створювати та налаштовувати R+RStudio+Git+GitHub середовище
- Працювати з локальним і віддаленим репозиторіями за допомогою Git

---

Хід роботи

1. Інсталяція та налаштування Git

Перевіримо наявність Git у системі:

```
!git --version
```

Якщо Git не встановлено, виконуємо інсталяцію:

```
!apt-get update
!apt-get install -y git
```

2. Налаштування профілю користувача Git

Налаштуємо ім'я користувача та електронну пошту:

```
!git config --global user.name "Solomka Boris"
!git config --global user.email "boris.solomka@example.com"
```

Перевіримо налаштування:

```
!git config --list
```

3. Створення репозиторію на GitHub

Створено репозиторій на GitHub через веб-інтерфейс:

- Назва репозиторію: `myrepo`
- Тип: Public
- Ініціалізовано з README.md
- Додано .gitignore для R
- Ліцензія: MIT

URL репозиторію: `https://github.com/username/myrepo.git`

4. Ситуація I: Клонування віддаленого репозиторію

Клонуємо репозиторій з GitHub на локальну машину:

```
# Переходимо в робочий каталог
%cd /content

# Клонуємо репозиторій
!git clone https://github.com/username/myrepo.git
```

Переходимо до клонованого репозиторію та перевіряємо його стан:

```
%cd myrepo

# Перевіряємо поточну гілку
!git branch

# Перевіряємо зв'язок з віддаленим репозиторієм
```

```
!git remote -v
```

5. Створення тестового R-скрипта

Створимо простий R-скрипт, який виводить "Hello world" n разів:

```
# Тестовий скрипт для виведення Hello world
# Автор: Соломка Борис Олегович
# Група: КН-24-1

n <- 5

for (i in 1:n) {
  print(paste("Hello world", i))
}
```

```
[1] "Hello world 1"
[1] "Hello world 2"
[1] "Hello world 3"
[1] "Hello world 4"
[1] "Hello world 5"
```

6. Інсталяція R та виконання скрипта

Встановлюємо R:

```
!apt-get install -y r-base
```

Виконуємо створений R-скрипт:

```
!Rscript hello_world.R
```

7. Робота з Git: додавання файлів та коміт

Перевіримо статус репозиторію:

```
!git status
```

Додаємо файл до staging area:

```
!git add hello_world.R
```

Створюємо коміт:

```
!git commit -m "Додано тестовий R-скрипт для виведення Hello world"
```

8. Відправка змін на GitHub

Відправляємо коміт на віддалений репозиторій:

```
!git push origin main
```

9. Ситуація II: Зв'язування наявного локального репо з віддаленим

Створимо новий локальний репозиторій:

```
# Створюємо новий каталог
!mkdir -p /content/local_project
%cd /content/local_project

# Ініціалізуємо Git-репозиторій
!git init
```

Перевіримо зв'язок з віддаленими репозиторіями:

```
!git remote -v
```

Додаємо зв'язок з віддаленим репозиторієм:

```
!git remote add origin https://github.com/username/myrepo.git

# Перевіряємо
!git remote -v
```

10. Перегляд історії комітів

```
%cd /content/myrepo
!git log --oneline --graph --all
```

Відповіді на контрольні питання

1. Що таке система контролю версій (СКВ) і навіщо вона потрібна?

Відповідь:
Система контролю версій (Version Control System, VCS) — це програмне забезпечення, яке дозволяє відстежувати зміни у файлах проєкту з плином часу. СКВ потрібна для:

- Збереження історії всіх змін у коді
- Можливості повернення до попередніх версій
- Ефективної командної роботи над одним проєктом
- Відстеження, хто і коли вніс конкретні зміни
- Створення різних гілок розробки для експериментів
- Резервного копіювання коду

2. Яка різниця між Git та GitHub?

Відповідь:

- **Git** — це розподілена система контролю версій, програмне забезпечення, яке встановлюється локально на комп'ютері та дозволяє керувати версіями файлів.
- **GitHub** — це веб-сервіс для хостингу Git-репозиторіїв, який надає зручний інтерфейс для роботи з Git, а також додаткові функції для командної роботи (pull requests, issues, wiki тощо).

Простіше кажучи: Git — це інструмент, а GitHub — це платформа для зберігання та спільної роботи з Git-репозиторіями.

3. Що таке коміт (commit) у Git?

Відповідь:
Коміт — це "знімок" стану проєкту в певний момент часу. Кожен коміт містить:

- Зміни, внесені у файли
- Повідомлення, що описує ці зміни

- Повідомлення, що описує ці зміни
- Інформацію про автора та час створення
- Унікальний ідентифікатор (hash)

Коміти дозволяють зберігати історію розробки та за потреби повертатися до попередніх станів проєкту.

4. Поясніть команди: git add, git commit, git push

Відповідь:

- **git add** — додає файли до staging area (область підготовки). Це означає, що ми вказуємо Git, які зміни мають бути включені в наступний коміт.

```
git add filename.txt    # додати конкретний файл
git add .              # додати всі змінені файли
```

- **git commit** — створює коміт (зберігає зміни) з файлів, що знаходяться в staging area. Кожен коміт повинен мати описове повідомлення.

```
git commit -m "Опис змін"
```

- **git push** — відправляє локальні коміти на віддалений репозиторій (наприклад, на GitHub).

```
git push origin main
```

5. Що таке гілка (branch) у Git і навіщо вона потрібна?

Відповідь:
Гілка — це незалежна лінія розробки у репозиторії. За замовчуванням створюється головна гілка (main або master).

Призначення гілок:

- Розробка нових функцій без впливу на основний код
- Експерименти з кодом
- Виправлення помилок ізольовано від основної розробки
- Паралельна робота кількох розробників

Основні команди:

```
git branch              # переглянути гілки
git branch new-feature  # створити нову гілку
git checkout new-feature # перейти до гілки
git merge new-feature   # об'єднати гілку з поточною
```

6. Як клонувати репозиторій з GitHub?

Відповідь:
Для клонування репозиторію використовується команда `git clone`:

```
git clone https://github.com/username/repository.git
```

Ця команда:

1. Створює локальну копію віддаленого репозиторію
2. Автоматично налаштовує зв'язок з віддаленим репозиторієм (origin)
3. Завантажує всю історію комітів
4. Встановлює робочу директорію на головну гілку

7. Що означає команда git remote -v?

Відповідь:
Команда `git remote -v` виводить список віддалених репозиторіїв, пов'язаних з локальним репозиторієм, разом з їх URL-адресами.

Параметри виводу:

- Назва віддаленого репозиторію (зазвичай `origin`)
- URL для fetch (завантаження)
- URL для push (відправки)

Приклад виводу:

```
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)
```

8. Як налаштувати зв'язок локального репозиторію з віддаленим?

Відповідь:
Для зв'язування локального репозиторію з віддаленим використовується

Для зв'язування локального репозиторію з віддаленим використовується команда:

```
git remote add <nickname> <url>
```

Приклад:

```
git remote add origin https://github.com/username/myrepo.git
```

Де:

- `origin` — стандартна назва для головного віддаленого репозиторію (можна використовувати будь-яку назву)
- URL — адреса віддаленого репозиторію на GitHub

Перевірка:

```
git remote -v
```

Висновок

Під час виконання лабораторної роботи було успішно опановано основи роботи з системою контролю версій Git та платформою GitHub.

Виконані завдання:

1. **Встановлено та налаштовано Git** — проведено інсталяцію Git, налаштовано профіль користувача з ім'ям та електронною поштою, що є необхідною умовою для роботи з системою контролю версій.
2. **Створено репозиторій на GitHub** — успішно створено публічний репозиторій з базовими налаштуваннями (README, .gitignore для R, ліцензія MIT).
3. **Опрацьовано дві типові ситуації роботи з Git:**
    - Клонування віддаленого репозиторію на локальну машину
    - Зв'язування існуючого локального репозиторію з віддаленим на GitHub
4. **Створено та виконано тестовий R-скрипт** — розроблено програму, яка виводить "Hello world" задану кількість разів, продемонстровано два методи реалізації (цикл for та функція replicate).

5. **Практично застосовано основні команди Git:**
    - `git init` — ініціалізація репозиторію
    - `git clone` — клонування репозиторію
    - `git add` — додавання файлів до staging area
    - `git commit` — створення коміту
    - `git push` — відправка змін на віддалений репозиторій
    - `git remote` — управління віддаленими репозиторіями
    - `git status`, `git branch`, `git log` — перегляд стану репозиторію
6. **Інтегровано R з Git** — налаштовано середовище для роботи з R-проєктами під контролем версій.

Практична значущість:

Отримані навички є фундаментальними для сучасної розробки програмного забезпечення та наукових досліджень. Система контролю версій Git дозволяє:

- Ефективно працювати в команді
- Відстежувати історію змін проєкту
- Безпечно експериментувати з кодом
- Створювати резервні копії роботи
- Документувати процес розробки

Використання GitHub як віддаленого сховища забезпечує доступність коду з будь-якого місця, можливість співпраці з іншими розробниками та створення портфоліо проєктів.

Лабораторна робота виконана повністю, всі поставлені цілі досягнуто.