

Архитектура компьютера Отчёт по лабораторной работе №2

Лю Сяо НКАбд-04-24

1 Описание задачи

Изучайте git. Завершите базовые настройки git и создайте библиотеку с помощью шаблона

2 Описание результатов выполнения задания

Войдите в GitHub, чтобы создать новую библиотеку и использовать шаблон.

New repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk ().*

Repository template

yamadharma/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ Include all branches
Copy all branches from yamadharma/course-directory-student-template and not just the default branch.

Owner * **Repository name ***

lsbigheader ▾ / student_2025_1x

✔ student_2025_ is available.

Great repository names are short and memorable. Need inspiration? How about **special-disco** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

You are creating a public repository in your personal account.

Create repository

Репозиторий: <https://github.com/yamadharma/course-directory-student-template>.

Установить информацию git

```
liveuser@localhost-live:~/work$ git config --global user.name "liuxiao"
liveuser@localhost-live:~/work$ git config --global user.email "1915054374@qq.com"
liveuser@localhost-live:~/work$ git config --list
user.name=liuxiao
user.email=1915054374@qq.com
liveuser@localhost-live:~/work$
```

Сгенерировать ssh-ключ

```
liveuser@localhost-live:~/work$ ssh-keygen -C '1915054374@qq.com' -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_rsa):
Created directory '/home/liveuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_rsa
Your public key has been saved in /home/liveuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0e8NSCm9399Fjbpe74CC9eaAZJ+RhLDMMgqF/FIClvM 1915054374@qq.com
The key's randomart image is:
+---[RSA 3072]-----+
|.+.o. o. |
|.o= . o.o |
| o= ..o |
|.Eo o + . ..|
| . . S = . o|
| = @ + o . |
| = B = o . |
| o O o +. |
| o.*...+ |
+---[SHA256]-----+
liveuser@localhost-live:~/work$ cd ~/.ssh/
liveuser@localhost-live:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCS9UrhHMmD7PF4P6AJqCLLLy+1GRffkwB5IELasv0F
ODKiWQIAKnWJujM04h7RdQ1tD5AKAIXtPQEt1k8R44+JI2AxBVvXE/1y02yVYBrca1Q8KyLe7Pr5I7Zl
6grtSgKVIuDfrXqwcTuoi77nK9xXHa64kTfV1bIH/TJQZumjGbnTpYa8AL3ih4PFc6pHNLrhMF4o9Qk7
VYCRswDrjD0NzuTAXrHluYAZq+CwhU0ffIwDz9V3lVJw67re1cR6q5JiilY/YmZUbyLPMKk+qTzK0h++
XmkFWFYyjG5MiR0C7a6LOf9PVnBiyyFXt4vVokS8Q0fLaQdMPzgRyugnJTzqP5n7vvC37eHCiJVnvwDJ
Mu+mDRDmMWGk1INtTDxB4puPFuRZqe39sI9wA8lsXmZj80esIxL4IKydaqy07Gwvsp2+ZRws+AgJu/Hf
AorSmWE0clWQ03/gvohw0vCdYRCa3XCuV4KXaBLvA/AA9NTl0QbBntLDhIpIUCH7xfHd+U= 1915054
374@qq.com
liveuser@localhost-live:~/.ssh$
```

Настройте ключ: скопируйте ключ на GitHub

Title

test

Key type

Authentication Key ↕

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGC9UrhHMmD7PF4P6AJqCLLly+1GRffkwB5IELa
sv0FODKiWQIAKnWJujMO4h7RdQ1tD5AKAIXtPQEt1k8R44+JI2AxBVvXE/1yO2yVYBrca1Q
8KyLe7Pr5I7Zl6grtSgKVluDfrXqwcTuoi77nK9xXHa64kTfV1bIH/TJQZumjGbnTpYa8AL3ih4P
Fc6pHNLrhMF4o9Qk7VYCRswDrjD0NzuTAXrHluYAZq+CwhU0fflwDz9V3IVJw67re1cR6q5
JiilY/YmZUbyLPMKk+qTzKOh++XmkFWFYjG5MiR0C7a6LOf9PVnBiyyFXt4vVokS8Q0fLaQ
dMPzgRyugnjTzqP5n7vvC37eHCiJVnwvDJMu+mDRDmMWGk1INtTDxB4puPFuRZqe39sl9
wA8lsXmZj8OeslxL4IKydaqy07Gwvsp2+ZRws+AgJu/HfAorSmWEd0clWQ03/gvohw0vCdY
RCa3XCUvK4XaBLvA/AA9NTIOQbBntLDhIplUCh7xfHd+U= 1915054374@qq.com
```

Add SSH key

Протяните проект через ssh `git clone git@github.com:Isbigheader/student_2025_lx.git`

```
$ git clone git@github.com:Isbigheader/student_2025_lx.git
Cloning into 'student_2025_lx'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (36/36), 19.38 KiB | 2.77 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

```
liveuser@localhost-live:~/work/student_2025_lx$ git pull
Already up to date.
liveuser@localhost-live:~/work/student_2025_lx$
```

Убедитесь, что файл обновлен.

Зафиксировать файлы в github git add . git commit -m "tijiao" git push

```
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git add .
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git commit -
m "tijiao"
[master d5d74ea] tijiao
21 files changed, 85 insertions(+)
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-1.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-10.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-11.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-12.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-13.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-14.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-15.png"
 create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\32
0\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-16.png"
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git push
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 4.70 MiB | 3.54 MiB/s, done.
Total 25 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Isbigheader/student_2025_lx.git
 d1092e4..d5d74ea master -> master
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$
```

Я завершил свою первую отправку задания

The screenshot shows a GitHub repository named 'student_2025_lx' (Public) generated from 'yamadharm/course-directory-student-template'. The repository has 1 Unwatch, 0 Forks, and 0 Stars. The file list on the left includes 'config', 'template', 'Домашнее задание/01' (highlighted with a red box), '.gitattributes', '.gitignore', '.gitmodules', 'CHANGELOG.md', 'COURSE', 'LICENSE', 'Makefile', 'README.en.md', 'README.git-flow.md', 'README.md', and 'package.json'. All files were committed '1 hour ago' except for 'Домашнее задание/01' which was committed '4 minutes ago' by user 'tjjiao'. The right sidebar shows repository details: 'About' (No description), 'Releases' (No releases published), 'Packages' (No packages published), 'Languages' (Shell 79.2%, Makefile 18.4%, Emacs Lisp 2.4%), and 'Suggested workflows' (Build, Configure).

3 Выводы

Я научился использовать git

4 Контрольные вопросы

Системы контроля версий (VCS)

Системы контроля версий (VCS) — это программные инструменты, которые помогают управлять изменениями в исходном коде или других файлах. Они предназначены для решения следующих задач:

- Отслеживание изменений в файлах.
- Сохранение истории изменений.
- Облегчение совместной работы над проектами.
- Возможность отката к предыдущим версиям файлов.
- Управление параллельными версиями проекта (ветками).

Основные понятия VCS и их отношения

1. Хранилище (Repository):

- Это база данных, где хранятся все версии файлов, история изменений и метаданные. Хранилище может быть локальным или удалённым.

2. Commit (Фиксация):

- Это операция сохранения изменений в хранилище. Каждый коммит имеет уникальный идентификатор, сообщение и ссылку на предыдущий коммит, что формирует историю изменений.

3. История (History):

- Это последовательность коммитов, которая показывает, как файлы изменялись со временем. История позволяет отслеживать, кто, когда и какие изменения внес.

4. Рабочая копия (Working Copy):

- Это текущая версия файлов, с которыми работает пользователь. Рабочая копия может быть изменена, после чего изменения фиксируются в хранилище.

Централизованные и децентрализованные VCS

1. Централизованные VCS:

- Все изменения хранятся на центральном сервере. Пользователи работают с локальными копиями, но для фиксации изменений требуется подключение к серверу.
- Примеры: **SVN (Subversion)**, **CVS (Concurrent Versions System)**.

2. Децентрализованные VCS:

- Каждый пользователь имеет полную копию хранилища, включая всю историю изменений. Это позволяет работать автономно и синхронизироваться с другими репозиториями.
- Примеры: **Git**, **Mercurial**.

Действия с VCS при единоличной работе с хранилищем

1. Создание локального репозитория: `git init`.
2. Добавление файлов в отслеживание: `git add <файл>`.
3. Фиксация изменений: `git commit -m "Сообщение"`.
4. Просмотр истории изменений: `git log`.
5. Откат к предыдущей версии: `git checkout <хэш коммита>`.

Порядок работы с общим хранилищем VCS

1. Клонирование удалённого репозитория: `git clone <URL>`.
2. Создание новой ветки для работы: `git branch <имя ветки>`.

3. Переключение на ветку: `git checkout <имя ветки>`.
 4. Фиксация изменений: `git commit -m "Сообщение"`.
 5. Отправка изменений на удалённый репозиторий: `git push origin <имя ветки>`.
 6. Получение изменений от других разработчиков: `git pull`.
-

Основные задачи, решаемые Git

1. Управление версиями файлов.
 2. Организация совместной работы.
 3. Создание и управление ветками.
 4. Отслеживание изменений и истории.
 5. Разрешение конфликтов при слиянии.
-

Основные команды Git и их характеристики

1. **git init** — инициализация нового репозитория.
 2. **git clone** — клонирование удалённого репозитория.
 3. **git add** — добавление файлов в индекс для последующего коммита.
 4. **git commit** — фиксация изменений в репозитории.
 5. **git status** — просмотр состояния рабочей копии.
 6. **git log** — просмотр истории коммитов.
 7. **git branch** — управление ветками.
 8. **git checkout** — переключение между ветками или коммитами.
 9. **git merge** — слияние веток.
 10. **git pull** — получение изменений из удалённого репозитория.
 11. **git push** — отправка изменений в удалённый репозиторий.
 12. **git diff** — просмотр различий между файлами или коммитами.
-

Примеры использования Git

1. Работа с локальным репозиторием:

- Создание репозитория: `git init`.

```
liveuser@localhost-live:~/work$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/liveuser/work/.git/
```

- Добавление файла: `git add file.txt`.

```
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git add .
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git commit
```


- Коммит: `git commit -m "Добавлен file.txt".`

```
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git commit -m "tijiao"
[master d5d74ea] tijiao
21 files changed, 85 insertions(+)
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-1.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-10.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-11.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-12.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-13.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-14.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-15.png"
create mode 100644 "\320\224\320\276\320\274\320\260\321\210\320\275\320\265\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\01/image-16.png"
```

2. Работа с удалённым репозиторием:

- Клонирование: `git clone git@github.com:Isbigheader/student_2025_lx.git.`

```
$ git clone git@github.com:Isbigheader/student_2025_lx.git
Cloning into 'student_2025_lx'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (36/36), 19.38 KiB | 2.77 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

- Отправка изменений: `git push.`

```
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$ git push
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 4.70 MiB | 3.54 MiB/s, done.
Total 25 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Isbigheader/student_2025_lx.git
d1092e4..d5d74ea master -> master
liveuser@localhost-live:~/work/student_2025_lx/Домашнее задание/01$
```

Ветви (Branches)

Ветви — это отдельные линии разработки, которые позволяют работать над разными версиями проекта одновременно. Они полезны для:

- Разработки новых функций без влияния на основную версию.
- Исправления ошибок в изолированной среде.
- Параллельной работы нескольких разработчиков.

Пример:

- Создание ветки: `git branch feature.`
- Переключение на ветку: `git checkout feature.`

Игнорирование файлов при коммите

Для игнорирования файлов используется файл **.gitignore**. В него добавляются шаблоны файлов или каталогов, которые не должны отслеживаться Git. Пример:

```
# Игнорировать все .log файлы
*.log

# Игнорировать каталог build
/build/
```

Это полезно для исключения временных файлов, файлов конфигурации или скомпилированных бинарников.