

## 2. Основные концепции Git

Раздел познакомит с терминологией Git, технической реализацией основных концепций, и предоставит примеры использования базовых команд.

### 2.1. Система отслеживания изменений

Представим производственную линию завода, на которой каждый день есть один сменяемый дежурный. По мере выполнения каждой операции дежурный заносит информацию о ней в блокнот, а вечером переносит содержимое блокнота в журнал. В результате получается три источника информации о выполненных работах:

1. Фактическое состояние линии, где часть работ может быть не завершена
2. Список выполненных работ в блокноте дежурного
3. Журнал изменений, хранящий последовательную историю всех работ

Аналогично этому примеру, Git хранит три состояния файлов.

1. Рабочая директория (**working directory**)

Директория проекта, в которой осуществляется работа с файлами.

2. Индекс (**staging area**)

Перечень файлов, подготовленных к отправке в репозиторий.

3. Репозиторий (**repository**)

Финализированная последовательная история изменений.

### 2.2. Структура репозитория

Git изначально задумывался, как децентрализованная система, поэтому всё необходимые для полноценной работы репозитория хранятся непосредственно в директории проекта внутри поддиректории `.git`

#### 💡 Tip

Любой репозиторий — ни что иное, как набор директорий и файлов.

У каждого пользователя может содержаться полная или частичная копия репозитория, позволяющая работать в условиях полной автономности.

Создание репозитория осуществляется с помощью команды `git init` [↗](#)

```
$ mkdir /tmp/git  
$ cd /tmp/git
```

```
$ git init .  
Initialized empty Git repository in /tmp/git/.git/
```

### ❗ Tip

Точка в конце строки является указателем на текущую директорию. Вместо точки можно использовать путь к произвольной директории.

В результате выполнения команды создается структура из директорий и файлов, необходимых для функционирования репозитория.

```
$ tree -aF -I '*sample'  
.  
├── .git/  
│   ├── HEAD  
│   ├── config  
│   ├── description  
│   ├── hooks/  
│   ├── info/  
│   │   └── exclude  
│   ├── objects/  
│   │   ├── info/  
│   │   └── pack/  
│   └── refs/  
│       ├── heads/  
│       └── tags/  
9 directories, 4 files
```

### ❗ Tip

Git создает файлы шаблонов в директории `hooks`, однако в рамках курса они будут скрываться до момента, когда курс затронет их напрямую.

## 2.3. Операции в рабочей директории

В процессе внесения изменений в файлы в директории проекта состояние репозитория не меняется. В этом можно убедиться, создав файл и повторив операцию листинга файлов.

```
$ echo 'Arbitrary text' > file1.txt
```

```
$ tree -aF -I '*sample'
```

```
.
├── .git/
│   ├── HEAD
│   ├── config
│   ├── description
│   ├── hooks/
│   ├── info/
│   │   └── exclude
│   ├── objects/
│   │   ├── info/
│   │   └── pack/
│   └── refs/
│       ├── heads/
│       └── tags/
└── file1.txt
```

```
9 directories, 5 files
```

### ❗ Tip

Git не выполняет операции самостоятельно и не отслеживает состояние файловой системы. Все действия над репозиторием выполняются путем запуска команд.

## 2.4. Подготовка к сохранению

После завершения работ над файлом необходимо добавить его в индекс, тем самым пометив его к сохранению в репозиторий. Для этого используется команда `git add` [↗](#)

```
$ git add file1.txt
```

```
$ tree -aF -I '*sample'
```

```
.
├── .git/
│   ├── HEAD
│   ├── config
│   ├── description
│   ├── hooks/
│   ├── index
│   ├── info/
│   │   └── exclude
│   ├── objects/
│   │   ├── 8e/
│   │   │   └── 4b04b468c5350cad218004489cc896e80df946
│   │   ├── info/
│   │   └── pack/
│   └── refs/
│       ├── heads/
│       └── tags/
└── file1.txt
```

В результате запуска команды в директории `.git` появились два файла:

- `.git/index`
- `.git/objects/8e/4b04b468c5350cad218004489cc896e80df946`

Первый файл является списком, в котором содержится перечень всех файлов, помеченных к сохранению. Для просмотра индекса используется команда

```
git ls-files --stage
```

```
$ git ls-files --stage
100644 8e4b04b468c5350cad218004489cc896e80df946 0    file1.txt
```

В выводе команды присутствуют тип исходного файла, его имя, права доступа, а также идентификатор из символов шестнадцатеричной системы, который является хэш-суммой от содержимого исходного файла. Он совпадает с именем одного из файлов в директории `.git/objects`, внутри которого в формате `gzip` находится содержимое исходного. Убедиться в этом можно с помощью команды `git cat-file`. Первый запуск определит тип объекта, а второй отобразит содержимое.

```
$ git cat-file -t 8e4b04b468c5350cad218004489cc896e80df946
blob
```

```
$ git cat-file blob 8e4b04b468c5350cad218004489cc896e80df946
Arbitrary text
```

### Tip

`blob` расшифровывается как `binary large object`

### Tip

Первые два символа хэш-суммы переносятся в имя директории с целью разделения файлов на группы. Это предотвращает замедление файловой системы.

## 2.5. Отслеживание состояния рабочей директории

Рассмотренная ранее команда `git ls-files` [↗](#) позволяет просматривать не только список подготовленных для сохранения файлов, но и перечень модифицированных или созданных файлов, не добавленных в индекс. Команда `git status` [↗](#) отображает все три набора файлов, чем существенно упрощает отслеживание изменений.

Создадим в директории еще один файл и рассмотрим вывод этой команды.

```
$ echo 'Arbitrary text' > file2.txt
```

```
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   file1.txt

Untracked files:
(use "git add <file>..." to include in what will be committed)
    file2.txt
```

Команда сравнивает состояние рабочей директории с индексом и сохраненными в репозитории данными, позволяя тем самым отслеживать появление новых файлов, удаление существующих и внесение изменений как до, так и после добавления файлов в индекс.

### Tip

Команда `git status` [↗](#) подсказывает синтаксис команд для работы с индексом.

Добавим второй файл в индекс и сравним вывод команд.

```
$ git add file2.txt
```

```
$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   file1.txt
    new file:   file2.txt
```

```
$ tree -aF -I '*sample'
.
├── .git/
│   ├── HEAD
│   ├── config
│   ├── description
│   ├── hooks/
│   ├── index
│   ├── info/
│   │   └── exclude
│   ├── objects/
│   │   ├── 8e/
│   │   │   └── 4b04b468c5350cad218004489cc896e80df946
│   │   ├── info/
│   │   └── pack/
│   └── refs/
│       ├── heads/
│       └── tags/
├── file1.txt
└── file2.txt

10 directories, 8 files
```

Не смотря на то, что файл был добавлен в индекс, количество файлов в директории `.git/objects` не изменилось. Это связано с тем, что Git не хранит метаинформацию внутри объекта и хэширует только содержимое файла.

### Tip

Хэширование содержимого без метаданных позволяет экономить пространство на диске и сокращать количество сохраняемых файлов.

## 2.6. Сохранение данных в репозиторий

Для того, чтобы перманентно сохранить текущее содержимое индекса в репозиторий, используется команда `git commit` [↗](#) Выполнение команды запустит текстовый редактор и предложит изменить сообщение, которое будет добавлено к сохраняемым

данным.

Содержание сообщения должно отражать суть вносимых изменений. Это может быть идентификатор задачи, упоминание исправляемой проблемы, общее описание правок. Git прервет выполнение операции, если сообщение будет пустым.

#### Tip

Большинство систем визуализации отображают только первую строку сообщения. Старайтесь делать ее содержимое кратким и емким.

#### Tip

Комментарий можно внести из командной строки: `git commit -m '<сообщение>'` ↗

После сохранения и закрытия текстового файла Git продолжит выполнять операции и выведет информацию по результатам.

```
$ git commit -m 'Add file1 and file2'
[master (root-commit) 6049234] Add file1 and file2
2 files changed, 2 insertions(+)
create mode 100644 file1.txt
create mode 100644 file2.txt
```

#### Tip

Объект, создаваемый в результате операции, также называется `commit`

Рассмотрим содержимое директории `.git` после выполнения данной операции.

```
$ tree -aF -I '*sample'
```

```
.
├── .git/
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ├── config
│   ├── description
│   ├── hooks/
│   ├── index
│   ├── info/
│   │   └── exclude
│   ├── logs/
│   │   ├── HEAD
│   │   └── refs/
│   │       └── heads/
│   │           └── master
│   ├── objects/
│   │   ├── 60/
│   │   │   └── 492349e637ccee64ceded6894b88f8e7e8bd72
│   │   ├── 8e/
│   │   │   └── 4b04b468c5350cad218004489cc896e80df946
│   │   ├── e7/
│   │   │   └── 4a8bdc9580ebed872e6b64a16c075f8194b1db
│   │   ├── info/
│   │   └── pack/
│   └── refs/
│       ├── heads/
│       │   └── master
│       └── tags/
├── file1.txt
└── file2.txt
```

15 directories, 14 files

В результате выполнения команды в директории `.git` появилось несколько новых файлов. Остановимся на файлах, размещенных в директории `.git/objects`. Как можно заметить, первые символы одного из них совпадают с набором символов в выводе команды `git commit` ↗, для просмотра его содержимого используем команду `git cat-file` ↗

```
$ git cat-file -t 60492349e637ccee64ceded6894b88f8e7e8bd72
commit
```

```
$ git cat-file commit 60492349e637ccee64ceded6894b88f8e7e8bd72
tree e74a8bdc9580ebed872e6b64a16c075f8194b1db
author Aleksei Sokolov <Aleksei_Sokolov2@epam.com> 1626338024 +0300
committer Aleksei Sokolov <Aleksei_Sokolov2@epam.com> 1626338024 +0300
```

Add file1 and file2



В тексте указан автор изменений, создатель коммита и соответствующие отметки времени. Также в первой строке отображается информация о новом объекте под названием `tree`. Его можно просмотреть с помощью команды `git cat-file` ↗. Объект `tree`, как и индекс, из которого он формируется, является бинарным. Для просмотра его содержимого необходимо конвертировать его в текстовый вид.

### ❗ Tip

Команда `git cat-file -p` ↗ приводит любой объект к текстовому виду.

```
$ git cat-file -t e74a8bdc9580ebed872e6b64a16c075f8194b1db
tree
```

```
$ git cat-file -p e74a8bdc9580ebed872e6b64a16c075f8194b1db
100644 blob 8e4b04b468c5350cad218004489cc896e80df946      file1.txt
100644 blob 8e4b04b468c5350cad218004489cc896e80df946      file2.txt
```



В дереве содержится перечень файлов с их типами, правами и идентификаторами объектов. Этот список связывает коммит с файлами, которые были добавлены в текущем коммите или существовали в репозитории до его появления, а также хранит метаданные всех перечисленных файлов.

### ❗ Tip

Git отслеживает смену имени, типа и прав доступа объектов файловой системы.

## 2.7. Итоги раздела

- Сущности репозитория хранятся внутри его собственной директории
- Git не производит автоматических операций над файловой системой
- Данные в Git разделены на три пространства
  - **рабочая директория** — фактическое состояние файлов в директории проекта
  - **индекс** — файлы, подготовленные к сохранению в репозиторий
  - **репозиторий** — файлы, сохраненные в виде наборов изменений
- Основные типы создаваемых объектов
  - **blob** — архивированное содержимое исходного файла
  - **tree** — дерево изменений с перечнем файлов и метаданных
  - **commit** — аннотированный набор изменений с указанием времени и авторства
- Команды для управления изменениями:
  - `git add` ↗ — добавление файлов в индекс
  - `git ls-files` ↗ — отображение списка файлов в пространствах
  - `git status` ↗ — вывод состояния изменений в репозитории

- `git cat-file`  — отображение содержимого объектов
- `git commit`  — сохранение изменений в историю репозитория