

Работа с репозиторием

Базовые команды

Основные темы

- Работа из командной строки
 - Основные команды
- Работа оффлайн
- Синхронизация
 - работа с удаленным репозиторием
- Работа с ветками
 - Построение workflow
- Разрешение конфликтов

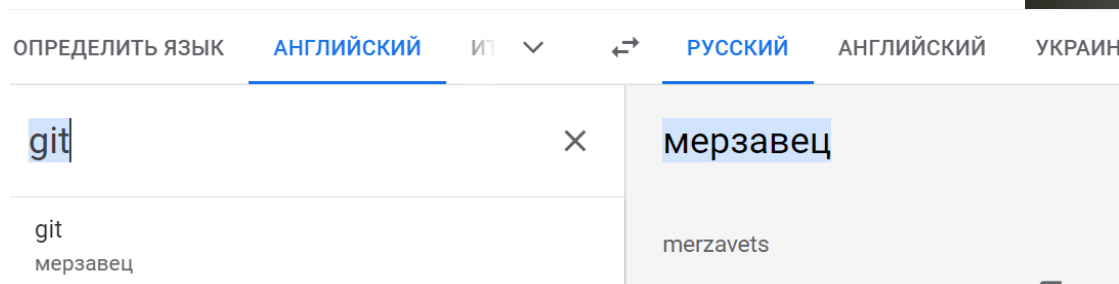
Git

- Распределенная система управления версиями
- Не первая, до нее были другие, такие, как
 - CVS - Concurrent Versions System
 - SVN – Subversion
- Но остался только он, git
 - Кстати, почему он так называется?

Git

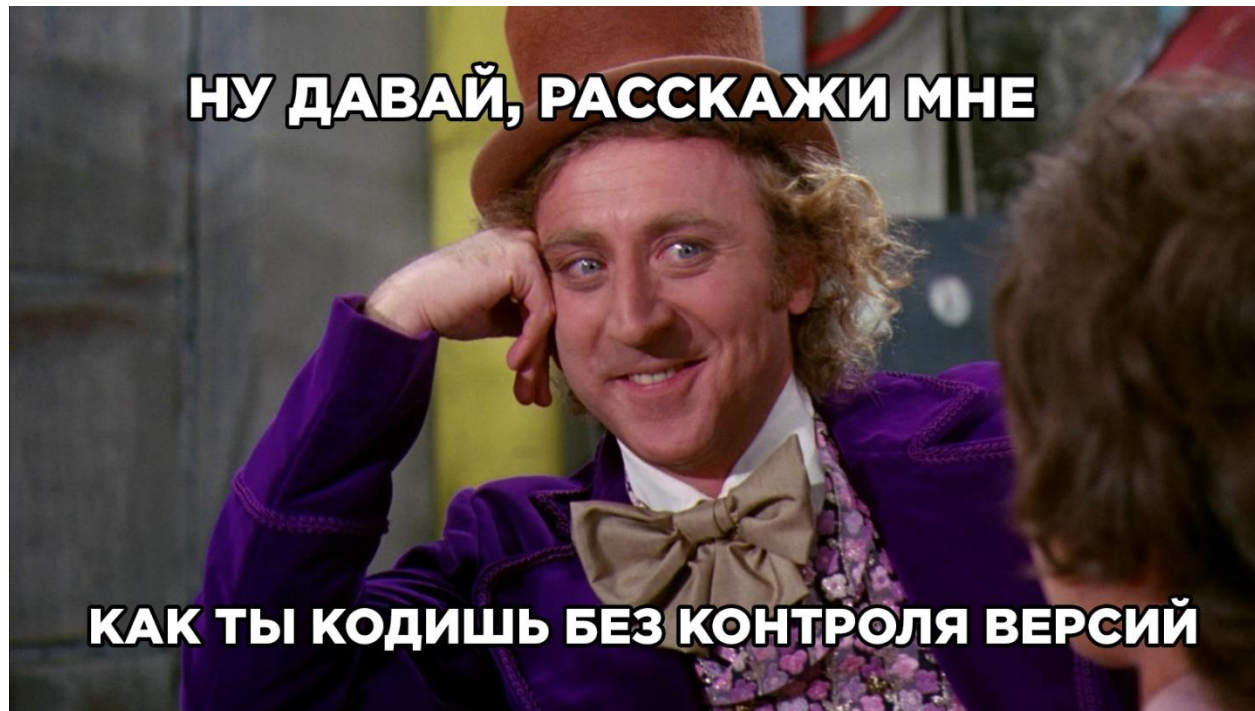
- “I'm an egotistical bastard, so I name all my projects after myself. First Linux, now git”

- Linus Benedict Torvalds



Git

- Работа с системой контроля версий – базовый навык любого разработчика
 - И не только разработчика, т.к. теперь вокруг нее построены DevOps, GitOps и т.п.
- Необходимый навык при командной работе над проектом

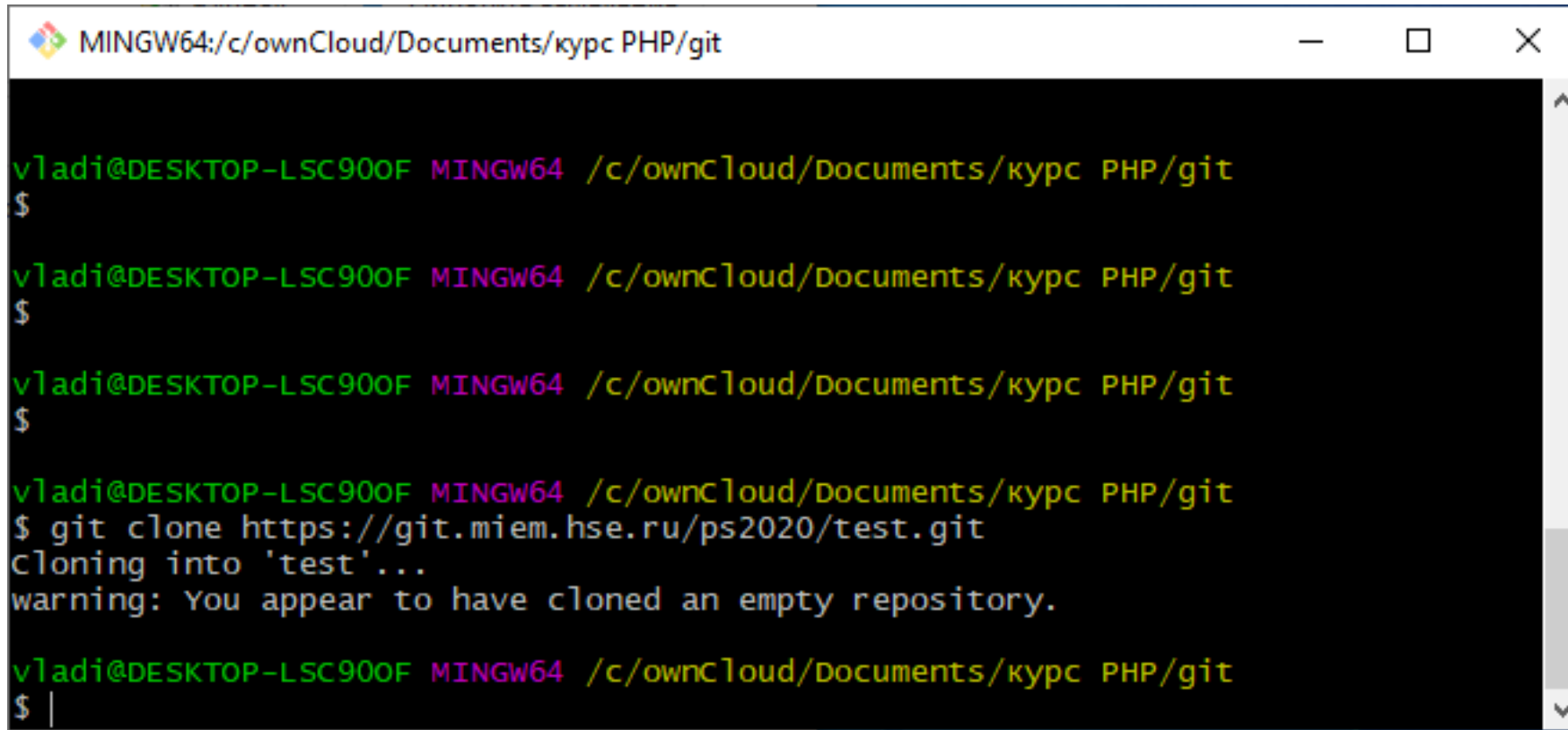


0. Создание репозитория

- Команда `git init`
- Создание через интерфейс gitlab
 - Можно сразу инициализировать репозиторий (сделать первый коммит)
- В момент создания репозитория
 - В корне проекта будет создана директория `.git`
 - Она содержит всю базу данных проекта и настройки
- Настройка текущего пользователя
 - `git config --global user.name "Name"`
 - `git config --global user.email vbashun@hse.ru`

0. Скачивание репозитория

- Делается однократно командой git clone



```
MINGW64:/c/ownCloud/Documents/курс PHP/git

v\ladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$

v\ladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$

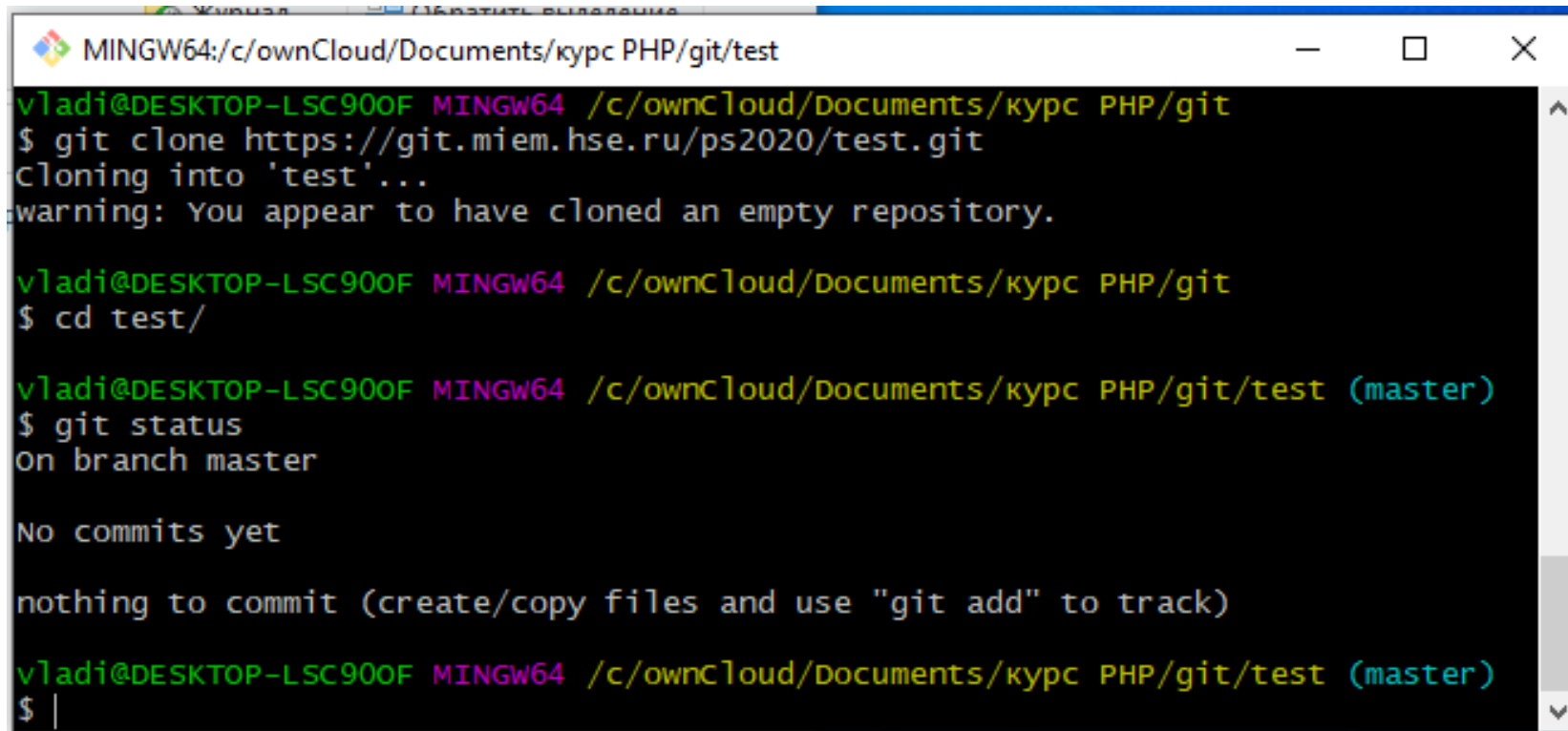
v\ladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$

v\ladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$ git clone https://git.miem.hse.ru/ps2020/test.git
Cloning into 'test'...
warning: You appear to have cloned an empty repository.

v\ladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$ |
```

1. Проверка статуса

- git status (в папке где находится .git)

A screenshot of a Windows command prompt window titled "MINGW64:/c/ownCloud/Documents/курс PHP/git/test". The window shows the following commands and output:

```
vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$ git clone https://git.miem.hse.ru/ps2020/test.git
Cloning into 'test'...
warning: You appear to have cloned an empty repository.

vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git
$ cd test/

vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git/test (master)
$ git status
On branch master

No commits yet

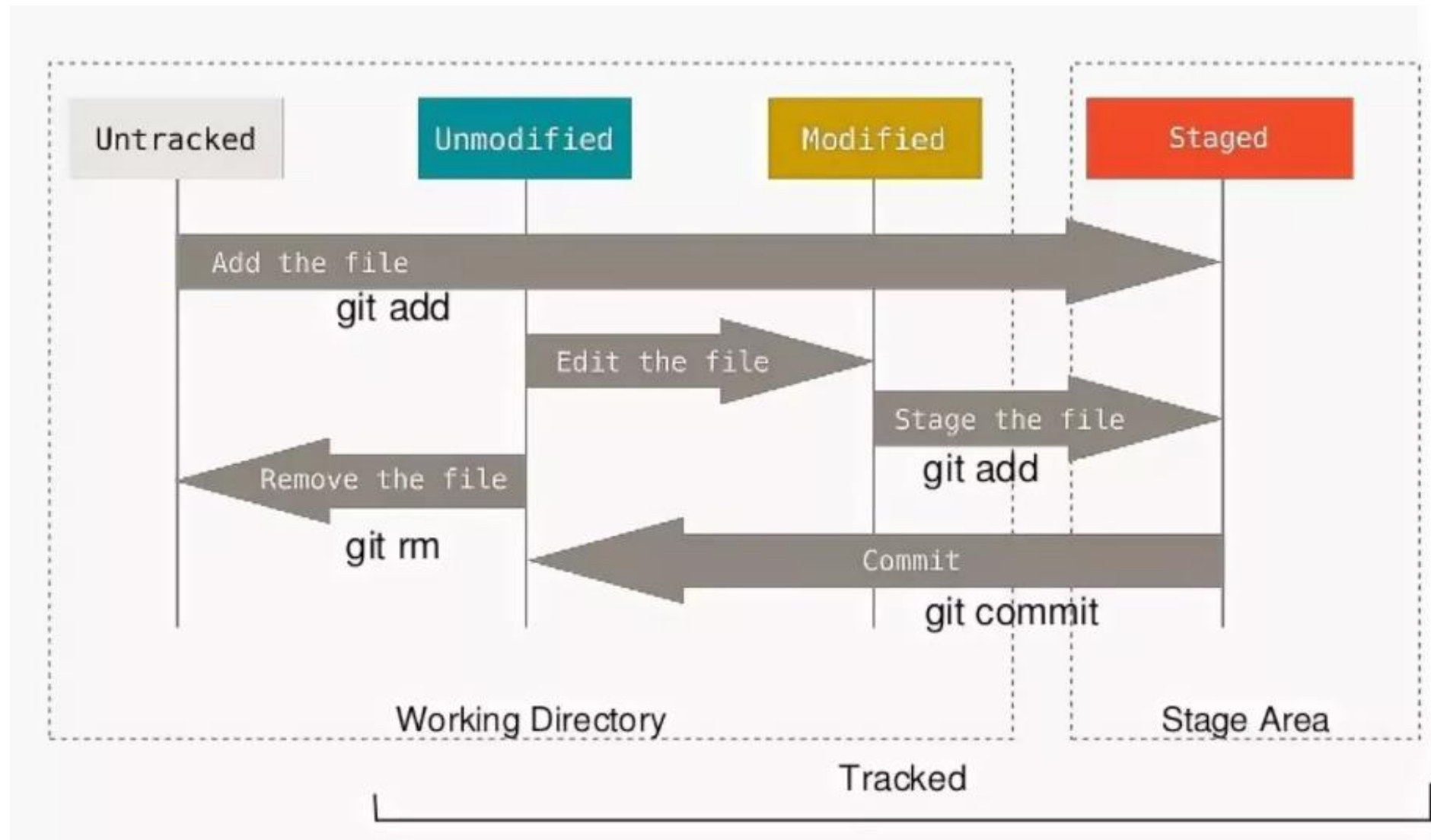
nothing to commit (create/copy files and use "git add" to track)

vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git/test (master)
$ |
```


2. Внесение изменений

- Добавляете изменения в исходные коды
- Повторное выполнение команды `git status` позволяет увидеть следующие категории файлов
 - Untracked - неотслеживаемые файлы, которые есть в директории, но еще не были добавлены в index (их нет в репозитории git)
 - Отслеживаемые, измененные – файлы, которые есть в репозитории git и при этом были изменены. Могут быть в двух состояниях:
 - Changes not staged for commit – еще не подготовлены к добавлению в репозиторий
 - Changes to be committed – подготовленные к добавлению в репозиторий
 - Отслеживаемые, неизмененные не будут отображаться этой командой

Статусы файлов git



Исходное состояние проекта

- git status (в папке где находится .git)

```
MINGW64:/c/ownCloud/Documents/кypc PHP/git/test
vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git
$ git clone https://git.miem.hse.ru/ps2020/test.git
Cloning into 'test'...
warning: You appear to have cloned an empty repository.

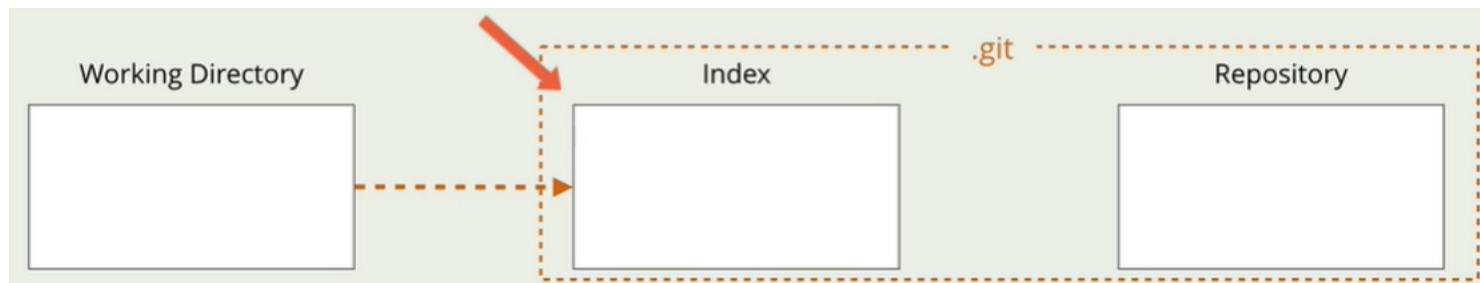
vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git
$ cd test/

vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ |
```



3. Перед stage

- Проверка изменений

```
MINGW64:/c/ownCloud/Documents/кypc PHP/git/test

nothing to commit (create/copy files and use "git add" to track)

vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git status
On branch master

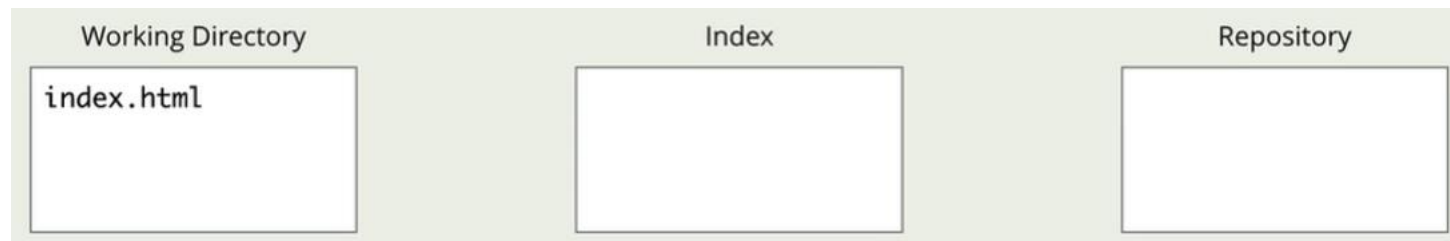
No commits yet

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

        task0/

nothing added to commit but untracked files present (use "git add" to track)

vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$
```



4. После stage

- Команда `git add`
- Git добавляет изменения в index
- Но они еще не сохранены в репозиторий

```
MINGW64:/c:/ownCloud/Documents/кypc PHP/git/test
vлади@DESKTOP-LSC900F MINGW64 /c:/ownCloud/Documents/кypc PHP/git/test (master)
$ git add task0/
vлади@DESKTOP-LSC900F MINGW64 /c:/ownCloud/Documents/кypc PHP/git/test (master)
$ git status
On branch master

No commits yet

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

    new file:   task0/solution.php

vлади@DESKTOP-LSC900F MINGW64 /c:/ownCloud/Documents/кypc PHP/git/test (master)
$ |
```



5. После commit

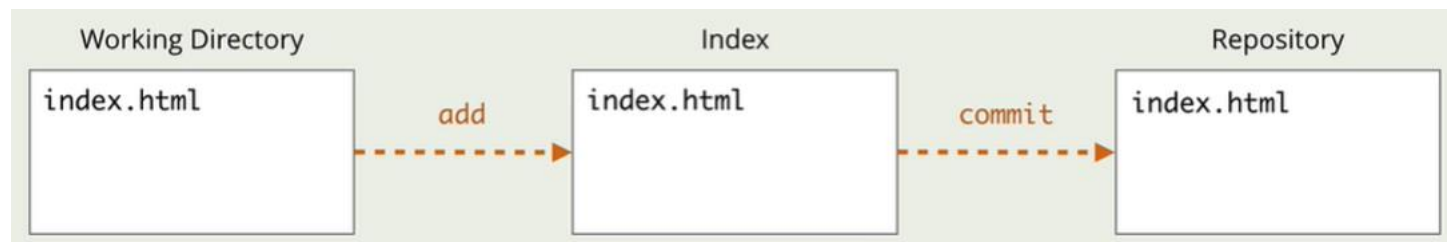
- Команда git commit
- Сохраняет изменения в репозиторий
- Важно – надо указать информацию о данном комите

```
vлади@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git commit -m "solution for task0"
[master (root-commit) 14/14d7] solution for task0
1 file changed, 3 insertions(+)
create mode 100644 task0/solution.php

vлади@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

vлади@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ |
```
























Стиль написания документов

- Хорошая практика – в начале сообщения указывать характер изменения в данном комите
 - feat – добавили новую возможность
 - test – поработали над тестированием
 - build – над сборкой
 - refactor – рефакторинг кода (изменение кода , например с целью его более понятного написания, без изменения функциональности этого кода)
 - fix – исправление ошибок в коде
 - WIP: неоконченная работа (хотя так делать не надо)
- Затем указывается компонент – часть проекта, с которым была работа и дополнительная информация

Commits · octokit/rest.js

GitHub, Inc. [US]https://github.com/octokit/rest.js/commits/master?after=96a2f9544c46ea974ce9eab255cd8680db5d01d1+35

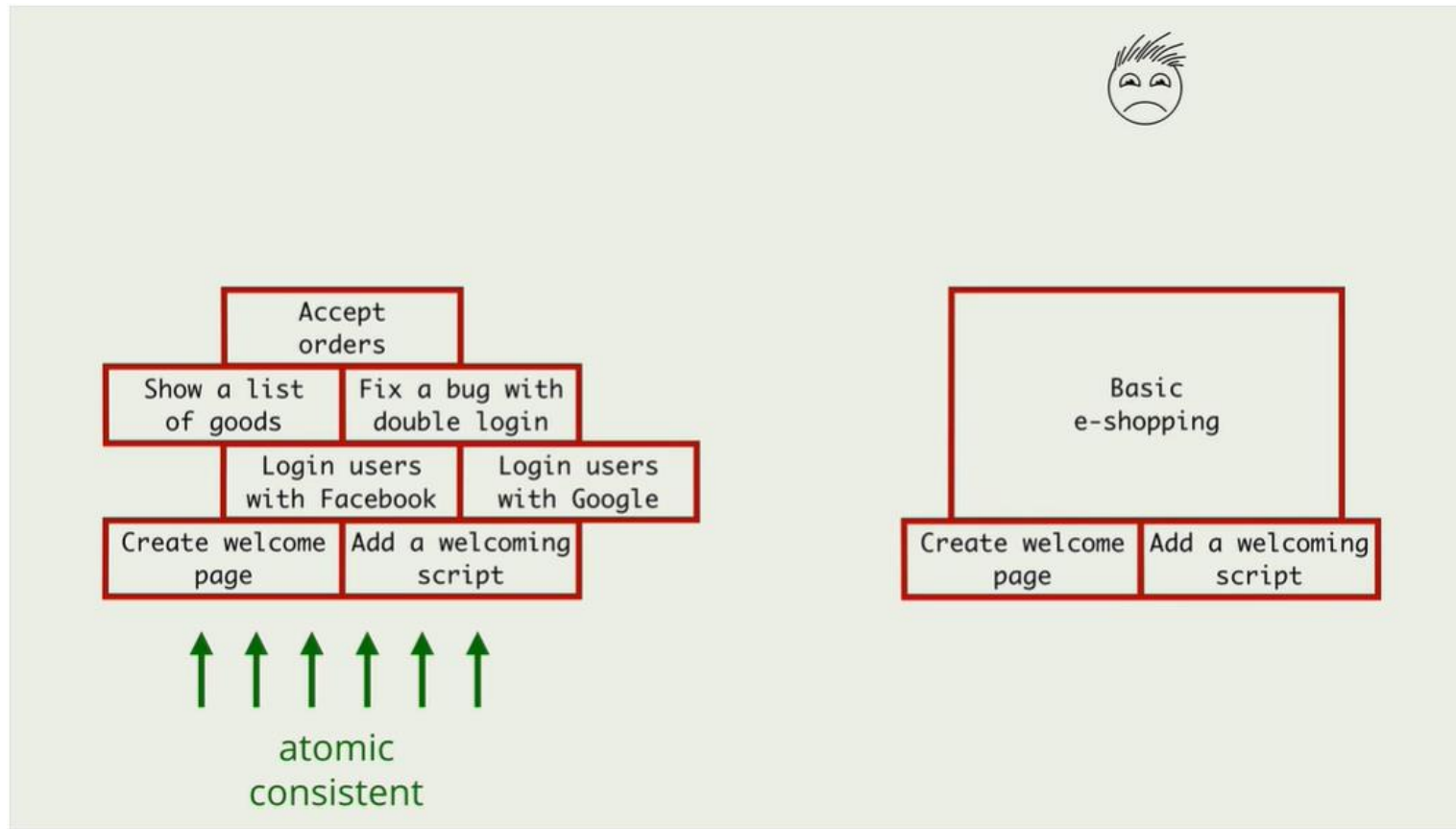
feat: support deprecation and aliasing of method names	 191a554 <>
 gr2m committed 22 days ago	
test: deprecate client.activity.getEventsForRepoIssues()	 a428fa7 <>
 gr2m committed 22 days ago	
build(routes-for-api-docs): initial version	 2f35924 <>
 gr2m committed 22 days ago	
test: error message values are wrapped by quotes if strings	 273ddce <>
 gr2m committed 24 days ago	
refactor(routes): rename parameters for type="object[]"	 ec4bec8 <>
 gr2m committed 24 days ago	
fix: validate "parameter[].property" values	 eefc826 <>
 gr2m committed 24 days ago	
test: validate "parameter[].property" values	 33ba6cc <>
 gr2m committed 24 days ago	
build(generate-api-docs): add link to V3 docs	 73ce0e8 <>
 gr2m committed 25 days ago	
build(generate-api-docs): adapt for new JSON	 a0087b6 <>
 gr2m committed 25 days ago	
refactor(routes): uploadAsset header parameter mapping	 9f795ce <>
 gr2m committed 26 days ago	
fix(validate): handle nested properties in endpoint definitions	 ... <>

Как часто делать коммиты?

- Хороший коммит – тот, который фиксирует одно небольшое законченное изменение
 - Такой коммит будет атомарным
- При этом он должен быть завершённым
 - Т.е. переводить систему из одного консистентного состояния в другое
 - Т.е. при «влипании» этого коммита, система не развалится, а при «откате» вернется в первоначальное состояние
- Коммит должен быть консистентным - логически завершённым изменением проекта

Как часто делать коммиты?

- Примеры разных подходов в коммитах



Как часто делать коммиты?

- Чем плох «мега-коммит»
 - Его сложнее понять другим
 - Сложнее отлавливать баги (не получится делать это по частям)
 - Сложнее откатиться назад (приходится откатывать все изменения)
 - Больше вероятность поймать конфликт при слиянии
 - Больше вероятность «сломать» другие части системы

Зачем вообще нужен индекс?

- Почему нельзя сразу делать коммит в репозиторий, зачем лишняя операция?
 - почему нельзя сделать просто – сделал – сохранил
- На самом деле, это удобно, и вот почему
 - В рабочей директории может накапливаться большое количество изменений
 - Если все эти изменения придется заливать одним коммитом, это не позволит следовать правилам «хороших коммитов» (атомарность, консистентность, одно законченное изменение на коммит и т.п.)
- Использование промежуточного stage позволяет выбирать файлы для текущего коммита

Working Directory

```
...  
index.html  
...  
auth/user.js  
...  
payment/card.js  
payment/refund.js
```

Index

```
...
```

Repository

```
...
```

Working Directory

```
...  
index.html  
...  
auth/user.js  
...  
payment/card.js  
payment/refund.js
```

add

Index

```
...  
index.html  
auth/user.js
```

Repository

```
...
```



Можно сделать коммит сразу без stage

- Команда `git commit` с флагом `-a`
 - изменения будут внесены в индекс, и тут же добавлены в репозиторий через новый коммит
 - При этом новые (не отслеживаемые) файлы не будут добавлены
- `git commit -a -m "feat: new file"`

5. После commit

- **ВАЖНО:** Пока что изменения залиты только в локальный репозиторий
 - На удаленный сервер их надо подгрузить отдельной командой
 - Помним, что гит позволяет работать и делать коммиты даже без подключения к серверу
 - Но в итоге, изменения надо залить на сервер
- Для работы с сервером есть две основные команды
 - push – отправить изменения на сервер
 - pull – подгрузить изменения с сервера
- Важно не забывать, что локальная версия репозитория может устаревать и расходиться с той версией, которая хранится на сервере

Заливка на сервер

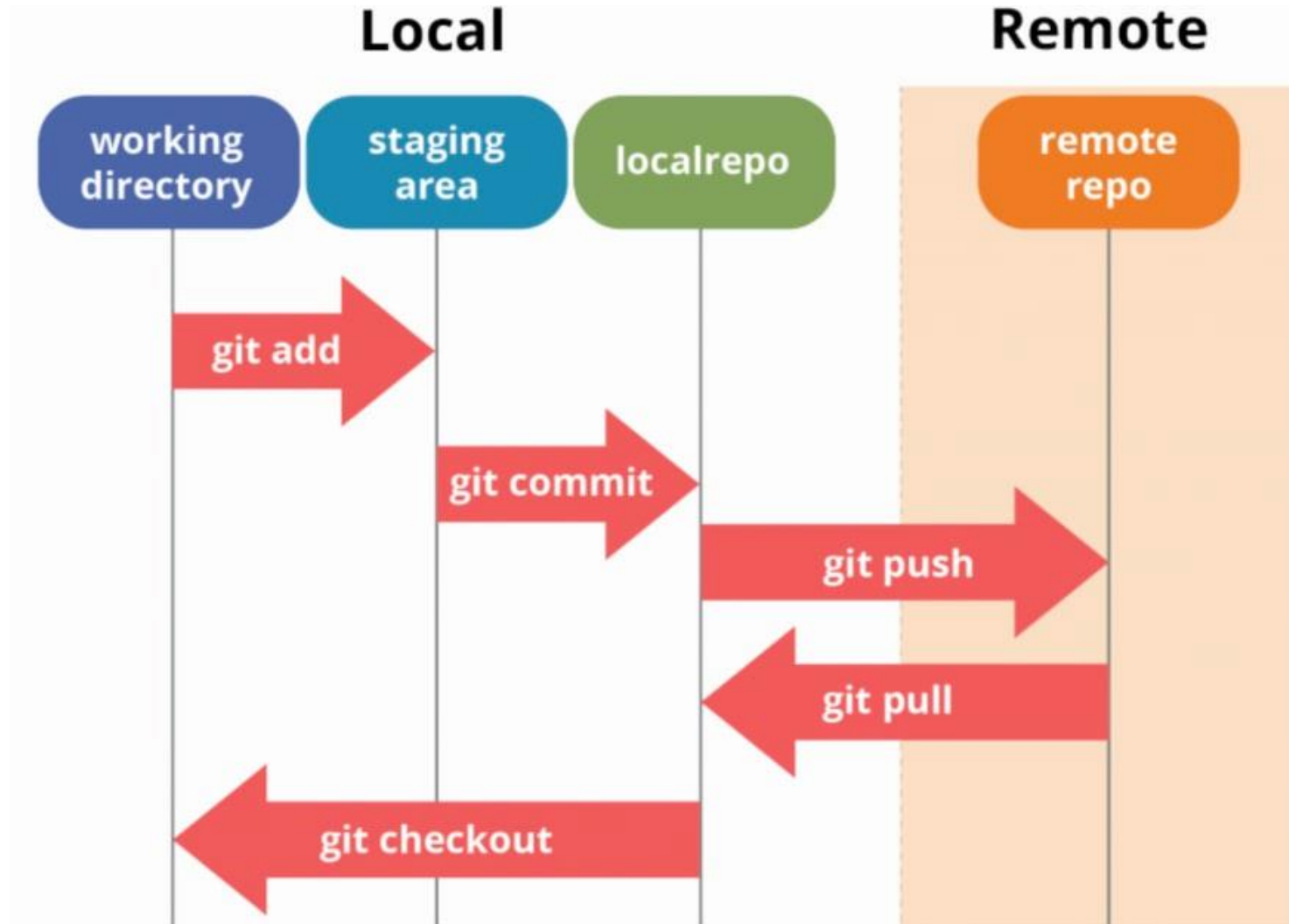
- Команда git push

```
vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
writing objects: 100% (4/4), 285 bytes | 285.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://git.miem.hse.ru/ps2020/test.git
 * [new branch]      master -> master

vldi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

git commands and statuses



5. Push

- Изменения будут залиты в ветку, которая соответствует текущей локальной ветке
 - Хотя мы еще не разбирались с ветками, у нас только master
 - Как работать с ветками, мы рассмотрим более подробно чуть позже
- Версия на сервере могла уже измениться по сравнению с той, что хранится у нас локально
 - Например, кто-то успел «вливать» свои изменения на сервер
 - В этом случае, будет ошибка при попытке сделать push

Проблемы при заливке (1)

- Команда `git push` не срабатывает, если версия на сервере более новая
- Необходимо сначала подгрузить обновления, используя `git pull`

```
vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git/test (master)
$ git push
To https://git.miem.hse.ru/ps2020/test.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://git.miem.hse.ru/ps2020/test.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

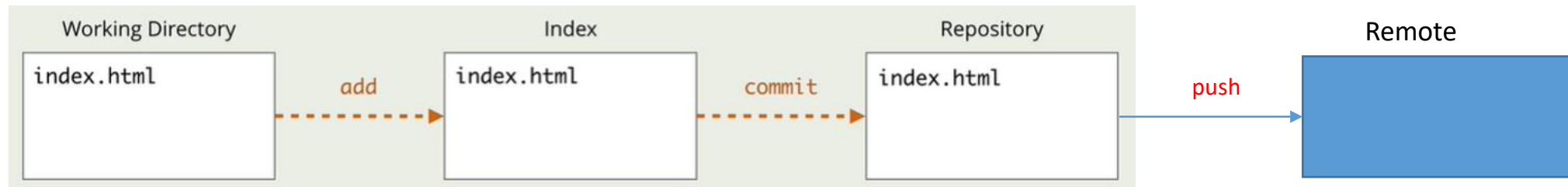
vladi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/курс PHP/git/test (master)
$ |
```

Проблемы при заливке (2)

- Сначала git pull, потом git push

```
v1adi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://git.miem.hse.ru/ps2020/test
   99ed685..9273169  master    -> origin/master
Merge made by the 'recursive' strategy.
 README.md | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 README.md

v1adi@DESKTOP-LSC900F MINGW64 /c/ownCloud/Documents/кypc PHP/git/test (master)
$ |
```



5. Push

- Проверка появления обновлений на сервере

The screenshot displays the GitLab interface for the 'cabinet-back' repository. The left sidebar shows the 'Commits' tab selected. The main area shows the commit history for the 'development' branch. The commits are listed as follows:

Commit Hash	Message	Author	Time Ago	Status
b16c4c3f	hotfix	Башун Владимир Владимирович	1 day ago	✓
9e93847b	hotfix: error on user creation	Башун Владимир Владимирович	1 day ago	✓
fd26d26e	Merge branch 'new-project-request-fields' into 'development'	Башун Владимир Владимирович	1 week ago	✓

Основные команды (самый простой flow)



Unstage

- Некоторые файлы не надо добавлять в гит
 - Это могут быть файлы конфигурации, файлы IDE и т.п.
 - Что делать, если мы их добавили, и как добиться того, чтобы они не добавлялись
- Допустим, мы случайно добавили файлы в index
 - Например, файлы .idea
- Как убрать эти файлы из index
- `git restore --staged <file>`
 - Уберет файл из индекса

```
D:\Универ\Летняя школа DevOps\test1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .idea/modules.xml
        new file:   .idea/test1.iml
        new file:   .idea/vcs.xml
```

.gitignore

- Можно явно указать git, что не надо добавлять некоторые файлы – чтобы он их игнорировал
 - Лучше сделать это сразу в начале работы, чтобы не решать проблемы с заливкой служебных файлов потом
- Сделать это просто – добавить файл .gitignore и указать те файлы или пути, которые git должен игнорировать
 - Можно указывать конкретные пути или паттерны
 - В каждой директории может быть свой

```
###> symfony/framework-bundle ###  
/.env.local  
/.env.local.php  
/.env.*.local  
/config/secrets/prod/prod.decrypt.private.php  
/public/bundles/  
/var/  
/vendor/  
###< symfony/framework-bundle ###  
.idea/*  
.php-cs-fixer.cache
```

.gitkeep

- Git будет игнорировать пустые директории
 - Т.е. команды `add` и `commit` не будут учитывать директорию, если в ней нет файлов
- Что делать, если нужно сохранить структуру или создать служебные каталоги?
- В нее надо что-то положить
 - Обычно просто пустой файл с названием `.gitkeep`



diff

- Перед тем, как добавлять изменения в index (делать stage файла), можно проверить, что в нем было изменено
- Для этого существует команда `git diff <filename>`
- Пример
 - Выполняем `git status`
 - Видим, что был изменен файл `hello.html`
 - Делаем команду `git diff hello.html`

```
$ git diff
diff --git a/hello.html b/hello.html
index cfd0a89..3808e2d 100644
--- a/hello.html
+++ b/hello.html
@@ -9,5 +9,7 @@
         <p>fff</p>
         test test test
         222
+       eeeee
+       eeee
</body>
</html>
```

Основные команды

- Создание репозитория
 - `git init`
- Клонирование удаленного репозитория с сервера
 - `git clone`
- Настройка пользователя
 - `git config --global user.name "Name"`
 - `git config --global user.email vbashun@hse.ru`
- Проверка статуса состояния репозитория
 - `git status`
- Добавление изменений в индекс
 - `git add .`
 - `git add filename`

Основные команды

- Фиксирование изменений
 - `git commit -m "doc: commit comment"`
 - `git commit -a -m "feat: new file"`
- Заливка изменений на сервер
 - `git push`
- Подгрузка обновлений с сервера
 - `git pull`
- Unstage file
 - `git restore --staged <file>`
- Просмотр изменений в файлах
 - `git diff <filename>`

git commands and statuses (extended)

