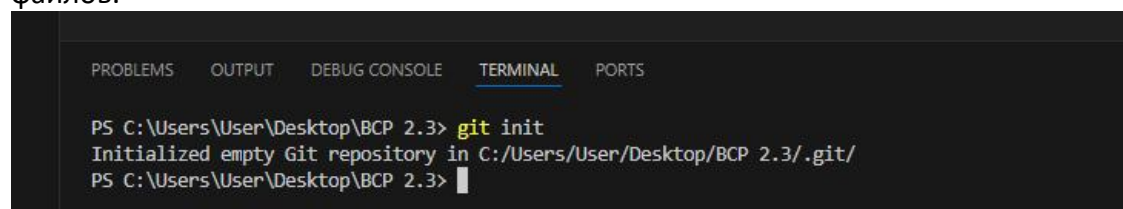


BCP 2.3. Встроенные средства IDE

Работа с командами Git: Visual Studio Code

1. Инициализация репозитория

Для создания нового репозитория в системе контроля версий Git необходимо выполнить команду `git init`. Данная команда инициализирует новый репозиторий, создавая скрытую директорию `.git`, в которой хранится вся информация о версиях, объектах и ссылках, необходимых для отслеживания истории изменений в проекте. Эта директория служит основой для работы с системой контроля версий, обеспечивая сохранение всех данных о коммитах и состоянии файлов.

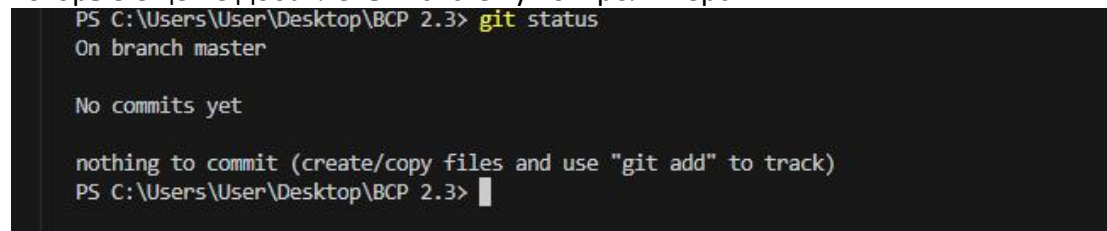


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\User\Desktop\BCP 2.3> git init
Initialized empty Git repository in C:/Users/User/Desktop/BCP 2.3/.git/
PS C:\Users\User\Desktop\BCP 2.3> 
```

2. Статус репозитория

Для получения информации о текущем состоянии репозитория следует использовать команду `git status`. Эта команда предоставляет сведения о подготовленных, неподготовленных и неотслеживаемых файлах. Она позволяет разработчикам быстро оценить, какие изменения были внесены, какие файлы находятся в стадии подготовки к коммиту, а также выявить новые файлы, которые еще не добавлены в систему контроля версий.

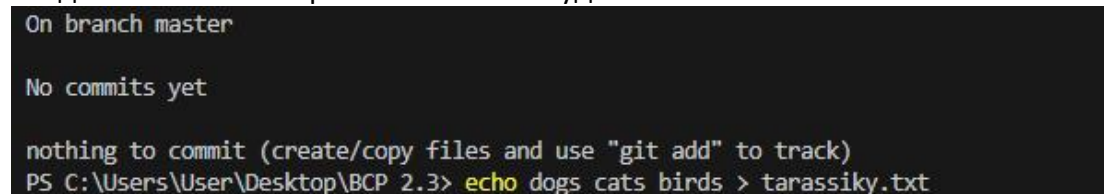


```
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\User\Desktop\BCP 2.3> 
```

Создание текстового файла с каким-нибудь текстом

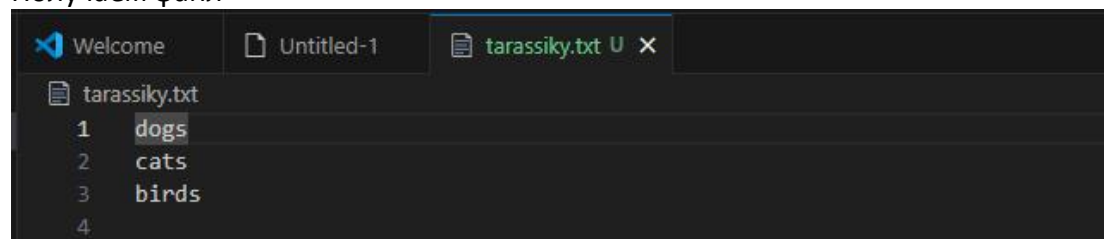


```
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\User\Desktop\BCP 2.3> echo dogs cats birds > tarassiky.txt
```

Получаем файл



```
Welcome  Untitled-1  tarassiky.txt U x

tarassiky.txt
1 dogs
2 cats
3 birds
4
```

3. Добавление файлов в область подготовленных файлов

Для добавления конкретного файла в область подготовленных файлов используется команда `git add`, за которой следует указание имени файла. После выполнения данной команды целевой файл будет помещен в индекс (область подготовленных файлов).

```
PS C:\Users\User\Desktop\BCP 2.3> git add .\tarassiky.txt
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master

No commits yet

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

PS C:\Users\User\Desktop\BCP 2.3> |
```

Рекомендуется сразу после этого проверить статус репозитория с помощью `git status`, чтобы убедиться, что файл был успешно добавлен в область подготовки.

4. Внесение изменений однострочным сообщением или через редактор

```
PS C:\Users\User\Desktop\BCP 2.3> git commit -m "Beautiful"
[master (root-commit) 38f2a3c] Beautiful
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 tarassiky.txt
PS C:\Users\User\Desktop\BCP 2.3> |
```

Для создания коммита с описанием изменений можно воспользоваться командой `git commit` с параметром `-m`, за которым следует текст сообщения в кавычках. Это позволяет документировать внесенные изменения непосредственно во время создания коммита.

И сразу проверяем статус репозитория

```
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\User\Desktop\BCP 2.3> |
```

Файл успешно зафиксирован

5. Просмотр истории коммитов с изменениями

Для анализа истории изменений в репозитории используется команда `git log`. Она отображает список последних коммитов в порядке их выполнения, что

позволяет разработчикам отслеживать прогресс проекта и изменения, внесенные на каждом этапе разработки. Добавление флага -p к команде git log позволяет получить более детальную информацию о внесенных изменениях в каждом файле, что способствует более глубокому пониманию эволюции проекта.

```
nothing to commit, working tree clean
PS C:\Users\User\Desktop\BCP 2.3> git log -p
commit 38f2a3c47dd938c310cb1251a17db2caf899667c (HEAD -> master)
Author: tarassiky <tarasova_dasha5@mail.ru>
Date: Sun Sep 22 11:45:18 2024 +0000

    Beautiful

diff --git a/tarassiky.txt b/tarassiky.txt
new file mode 100644
index 0000000..cf966cd
Binary files /dev/null and b/tarassiky.txt differ
PS C:\Users\User\Desktop\BCP 2.3>
```

6. Добавление удаленного репозитория

Чтобы связать локальный репозиторий с удаленным, необходимо использовать команду git remote add, указав имя и URL удаленного репозитория. Это действие создает ссылку на удаленный репозиторий, что позволяет выполнять операции синхронизации между локальной и удаленной версиями проекта.

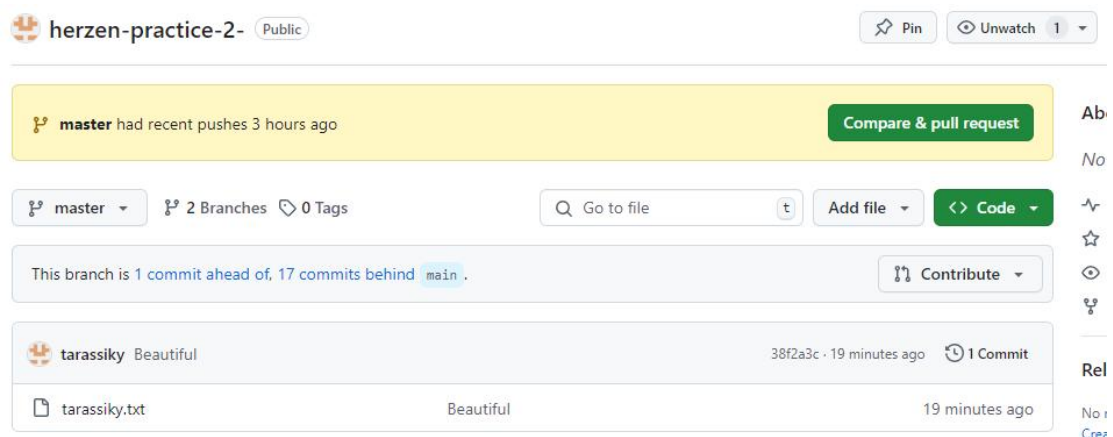
```
Binary files /dev/null and b/tarassiky.txt differ
PS C:\Users\User\Desktop\BCP 2.3> git remote add origin https://github.com/tarassiky/herzen-practice-2-
```

7. Отправка изменений в удаленный репозиторий

Для отправки локальных изменений в удаленный репозиторий используется команда git push, к которой добавляются имя удаленного репозитория и ветка, в которую будут отправлены изменения. Если отправляется новая ветка, необходимо использовать флаг -u, который устанавливает связь между локальной и удаленной веткой, упрощая последующие операции синхронизации.

```
PS C:\Users\User\Desktop\BCP 2.3> git remote add herzen-practice-2- https://github.com/tarassiky/herzen-practice-2-
PS C:\Users\User\Desktop\BCP 2.3> git push -u herzen-practice-2- master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 244 bytes | 244.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/tarassiky/herzen-practice-2-/pull/new/master
remote:
To https://github.com/tarassiky/herzen-practice-2-
 * [new branch]      master -> master
branch 'master' set up to track 'herzen-practice-2-/master'.
PS C:\Users\User\Desktop\BCP 2.3>
```

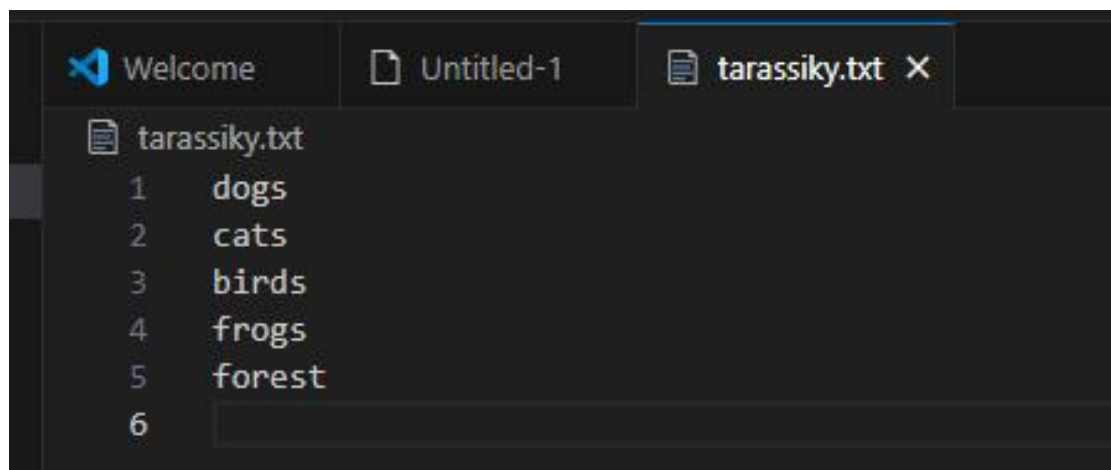
Далее, перейдя в GitHub, видим, что наш файл сохранился



8. Получение изменений из удалённого репозитория

Команда `git pull` применяется для загрузки изменений из удаленного репозитория. Она выполняет две операции: сначала загружает обновления с указанного удаленного репозитория, а затем объединяет их с текущей локальной веткой. Это позволяет поддерживать актуальность локальной копии проекта относительно его удаленной версии и гарантирует синхронизацию всех внесенных изменений.

```
PS C:\Users\User\Desktop\BCP 2.3> git pull origin master
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 1.70 KiB | 96.00 KiB/s, done.
From https://github.com/tarassiky/herzen-practice-2-
 * branch            master      -> FETCH_HEAD
 38f2a3c..9f96e86  master      -> origin/master
Updating 38f2a3c..9f96e86
Fast-forward
 tarassiky.txt | Bin 40 -> 37 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\User\Desktop\BCP 2.3> 
```



9. Просмотр изменений до коммита

Для анализа изменений, внесённых в репозиторий до их коммита, можно воспользоваться командой `git diff`. Этот инструмент позволяет детально рассмотреть список модификаций, произведённых с момента последнего коммита.

```
PS C:\Users\User\Desktop\BCP 2.3> git diff HEAD
diff --git a/tarassiky.txt b/tarassiky.txt
index 8f05b57..71c59d4 100644
--- a/tarassiky.txt
+++ b/tarassiky.txt
@@ -3,3 +3,4 @@ cats
 birds
 frogs
 forest
+bears
```

Например, если в файл `tarassiky.txt` была добавлена новая строка (`bears`), но в текущем состоянии репозитория этот файл не содержит указанной строки, то при выполнении команды `git diff` в терминале отобразятся все изменения, касающиеся данного файла.

Для просмотра изменений, которые были подготовлены к коммиту, следует использовать флаг `--staged`. Таким образом, если мы добавим файл `tarassiky.txt` в область подготовленных файлов с помощью команды `git add`, а затем проверим статус репозитория с помощью `git status`, мы увидим, что файл находится в состоянии "подготовлен к коммиту". Далее, применив команду `git diff --staged`, мы сможем увидеть изменения, которые будут зафиксированы в следующем коммите.

Перед применением команды я добавила в файл строку, а в репозитории этот же файл не содержит данную строку, поэтому в терминале мы можем увидеть изменения в файлах с момента последнего коммита.

```
PS C:\Users\User\Desktop\BCP 2.3> git add tarassiky.txt
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master
Your branch is ahead of 'herzen-practice-2-/master' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   tarassiky.txt
```

Для просмотра подготовленных изменений необходимо добавить флаг "--staged".

```
PS C:\Users\User\Desktop\BCP 2.3> git diff --staged
diff --git a/tarassiky.txt b/tarassiky.txt
index 8f05b57..71c59d4 100644
--- a/tarassiky.txt
+++ b/tarassiky.txt
@@ -3,3 +3,4 @@ cats
 birds
 frogs
 forest
+bears
```

10. Отмена подготовленных и неподготовленных изменений

Восстановление состояния подготовленного файла в рабочем дереве возможно с использованием команды `git reset`. Если после выполнения этой команды мы снова применим `git diff --staged`, терминал вернёт пустое значение, указывая на отсутствие подготовленных изменений.

```
PS C:\Users\User\Desktop\BCP 2.3> git reset tarassiky.txt
Unstaged changes after reset:
M      tarassiky.txt
PS C:\Users\User\Desktop\BCP 2.3> git diff --staged
PS C:\Users\User\Desktop\BCP 2.3>
```

Чтобы отменить неподготовленные изменения, можно воспользоваться командой `git checkout -- <имя_файла>`. Прежде чем выполнять эту операцию, рекомендуется проверить статус репозитория с помощью `git status`.

```
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master
Your branch is ahead of 'herzen-practice-2-/master' by 2 commits.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   tarassiky.txt

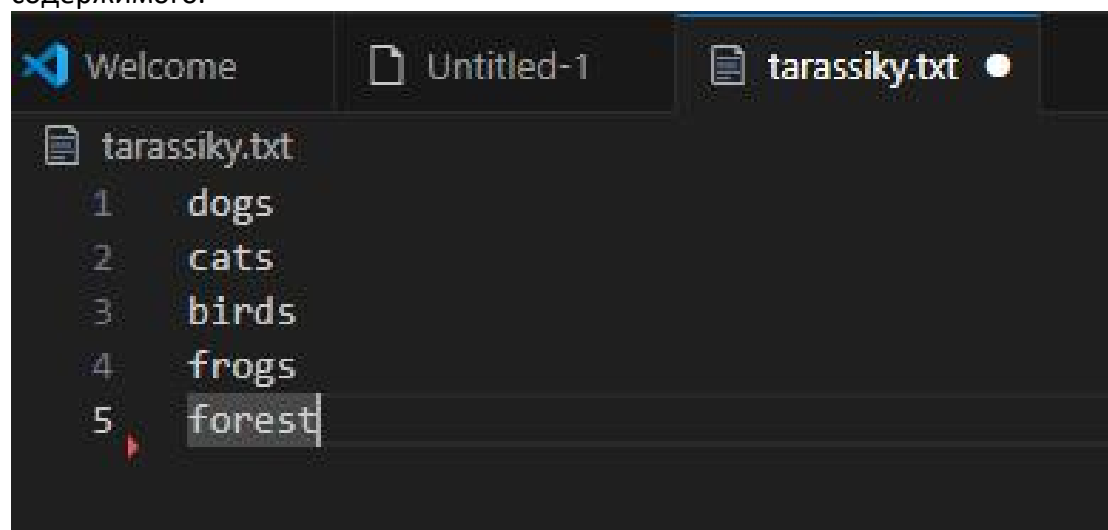
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\User\Desktop\BCP 2.3>
```

В результате выполнения команды `git checkout` изменённое состояние файла будет возвращено к состоянию последнего коммита.

```
PS C:\Users\User\Desktop\BCP 2.3> git checkout -- tarassiky.txt
PS C:\Users\User\Desktop\BCP 2.3> git status
On branch master
Your branch is ahead of 'herzen-practice-2-/master' by 2 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\User\Desktop\BCP 2.3> █
```

После этого, если мы снова проверим статус репозитория, изменённый файл `tarassiky.txt` перестанет отображаться, а слово "bears" исчезнет из его содержимого.



11. Создание новой ветки и переход в неё

Создание новой ветки осуществляется с помощью команды `git branch <имя_ветки>`. Однако Git не переключается на новую ветку автоматически. Для этого необходимо использовать команду `git checkout -b <имя_ветки>`, что создаст новую ветку и сразу же переключит на неё контекст.

```
PS C:\Users\User\Desktop\BCP 2.3> git checkout -b student
Switched to a new branch 'student'
PS C:\Users\User\Desktop\BCP 2.3> █
```

Чтобы определить текущую ветку, достаточно выполнить команду `git branch`, которая отобразит список всех веток с указанием активной.

```
PS C:\Users\User\Desktop\BCP 2.3> git branch
master
* student
PS C:\Users\User\Desktop\BCP 2.3> █
```

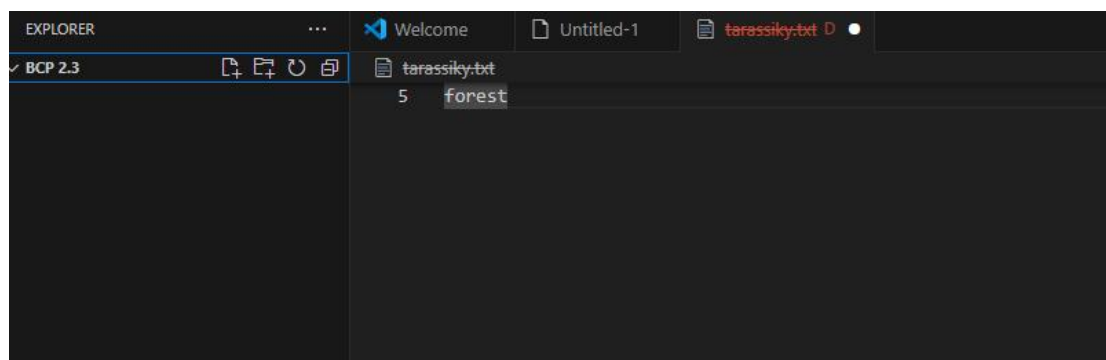
Переключение между существующими ветками осуществляется с помощью команды `git checkout <имя_ветки>` без дополнительных флагов. При создании новой ветки она наследует все файлы и изменения от ветки, на основе которой она была создана (например, от ветки `master`).

```
PS C:\Users\User\Desktop\BCP 2.3> git checkout master
Switched to branch 'master'
Your branch is ahead of 'herzen-practice-2-/master' by 2 commits.
(use "git push" to publish your local commits)
PS C:\Users\User\Desktop\BCP 2.3> █
```

12. Удаление файлов

Удаление файлов из текущего рабочего дерева выполняется с использованием команды `git rm <имя_файла>`. Также можно применять шаблоны для удаления нескольких файлов одновременно; например, использование маски `*.txt` приведёт к удалению всех файлов с расширением `.txt`. После выполнения команды для удаления файла `tarassiky.txt`, он будет исключён из рабочего дерева и подготовлен к коммиту.

```
PS C:\Users\User\Desktop\BCP 2.3> git rm .\tarassiky.txt
rm 'tarassiky.txt'
PS C:\Users\User\Desktop\BCP 2.3> █
```



Мы успешно удалили файл `tarassiky.txt`.

13. Слияние двух веток

Объединение двух веток осуществляется с помощью команды `git merge <имя_ветки>`, что приводит к слиянию указанных изменений с текущей активной веткой. Эта операция позволяет интегрировать изменения из одной ветки в другую, обеспечивая актуальность и согласованность кода.

```
PS C:\Users\User\Desktop\BCP 2.3> git merge student
Already up to date.
PS C:\Users\User\Desktop\BCP 2.3> █
```


14. Удаление ветки

Удаление ненужной ветки выполняется с использованием команды `git branch -d <имя_ветки>`. Перед выполнением данной операции рекомендуется проверить текущую активную ветку с помощью команды `git branch`. Например, если у нас есть две ветки — `master` и `student`, и мы решаем удалить ветку `student`, после выполнения команды удаления останется только ветка `master`. Это позволяет поддерживать порядок в репозитории и избегать накопления неиспользуемых веток.

```
PS C:\Users\User\Desktop\BCP 2.3> git branch
* master
  student
PS C:\Users\User\Desktop\BCP 2.3> git branch -d student
Deleted branch student (was 9f96e86).
PS C:\Users\User\Desktop\BCP 2.3> 
```

```
PS C:\Users\User\Desktop\BCP 2.3> git branch
* master
PS C:\Users\User\Desktop\BCP 2.3> 
```