

DevOps

2022.09.06

Opracował: Mikołaj Ziółek
ziolek.mikosz@protonmail.com

DevOps	2
Czym jest DevOps	2
Model C.A.M.S.	2
'The Three Ways'	3
Metodologia DevOps	4
Git Essentials	4
Basic commands	4
\$ git init	4
\$ git add	5
\$ git commit	5
git commit -- amend	5
\$ git log	6
\$ git status	7
\$ git diff	8
\$ git rm --cached	11
\$ git show	12
\$ git checkout	13
\$ git checkout -- <file_name>	13
\$git checkout <SHA> -- <file_name>	14
\$ git reset	16
\$ git revert	17
\$ git clean	18
Git Architecture	20
Hash Values & HEAD	21
Hash Value	21
H-E-A-D	22

DevOps

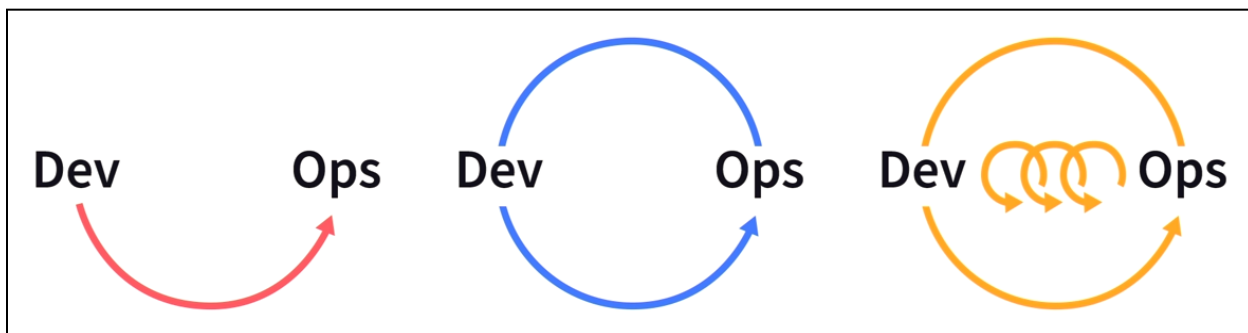
Czym jest DevOps

Albo raczej kim, jest to osoba wewnątrz projektu, która stara się ułatwić życie reszcie developerów poprzez m. in. **automatyzację procesów** przez nich wykonywanych, **nadzorowanie oprogramowania** lub samej **infrastruktury**, tworzenie pipeline'ów **CI/CD**.

Model C.A.M.S.

- **Culture**
 - Sedno tego czym zajmuje się DevOps jest związane z ludźmi i biznesem. Pomijając wszelkie zaawansowane narzędzia, z którym pracują, głównym zadaniem jest stworzenie odpowiedniej kultury w organizacji aby zapewnić rozwiązania dobrej jakości.
- **Automation**
 - Automatyzacją nazwiemy na przykład kawałek technologii, która wykonuje dane zadanie **bez** dodatkowej ingerencji człowieka. Celem tego zabiegu jest przyspieszenie i ułatwienie powtarzalnej pracy.
- **Measurement**
 - Metryka różnych aspektów będących częścią projektu może być pomocna, pomiary wartości t.j. *czas cyklu zadania, koszt, satysfakcja pracowników* mogą ułatwić znalezienie lepszego punktu widzenia na potencjalne problemy i zaangażować zespół, i ustalić ogólny cel.
- **Sharing**
 - ***"Discrete continuous improvement"***
 - *Dzielenie się wiedzą, pomysłami i nawet problemami jest bardzo ważne w życiu DevOps'a. Dzięki temu ktoś może się nauczyć czegoś nowego, lub ten ktoś udzieli Ci rady odnośnie problemu, z którym się zmagasz, lub zespołowi spodoba się Twój nowy pomysł i zostanie wdrożony.*

'The Three Ways'



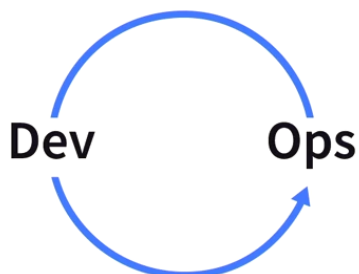
Systems Thinking



Ścieżka pierwsza mówi o tym, że na początku powinniśmy się skupić na **ogólnym wyniku** projektowego łańcucha wartości. Musimy pamiętać o tym, że optymalizacja jednego z elementów systemu, może skutkować pogorszeniem drugiego, a w końcu pogorszeniem się ogólnego wyniku.

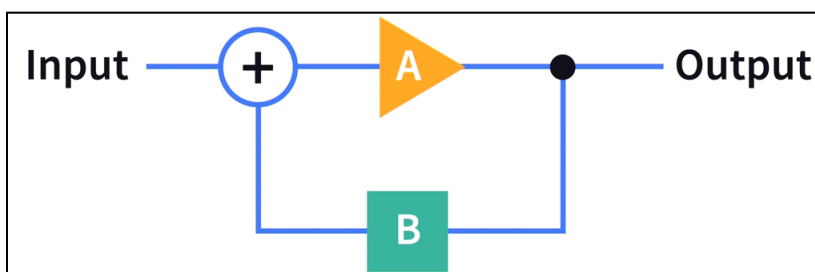
Np. Dodanie więcej serwerów do gry komputerowej, nie uwzględniając mocy przerobowej naszej bazy danych, która nie poradzi sobie z większą ilością zapytań może skutkować zawieszeniem serwera lub jego upadkiem.

Amplify Feedback Loops



Ścieżka druga nawiązuje do tworzenia, skracania i wzmacniania pętli pozyskiwania informacji zwrotnych (**feedback loop**) między częściami organizacji.

Feedback loop'em nazywamy proces, który bierze pod uwagę własne wyniki przy podejmowaniu decyzji, co dalej.



Continuous Experimentation



Ścieżka trzecia przypomina nam o tworzeniu kultury pracy, która pozwala zarówno na ciągłe eksperymentowanie, jak i uczenie się.

W tym podejściu skupiamy się na pracy nad sobą, doskonalenie umiejętności poprzez powtarzanie ćwiczeń, szukanie wyzwań i próbowanie nowych rozwiązań.

Metodologia DevOps

Git Essentials

Basic commands

```
$ git init
```

Po zainstalowaniu Git'a i utworzeniu naszego folderu projektowego możemy zainicjalizować nasze repozytorium przy pomocy komendy **git init**.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn  
$ git init  
Initialized empty Git repository in D:/Projects/devop-learn/.git/
```

Po wykonaniu komendy w folderze projektu zostanie utworzony folder **'git'**. To w nim odbywa się cała magia **śledzenia zmian**. Tutaj Git przechowuje wszystko, co wie o naszym projekcie.

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ ls -la
total 8
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:58 ./
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:57 ../
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:58 .git/
```

```
$ git add
```

Aby śledzić zmiany, należy najpierw jakichś dokonać. W tym przypadku w folderze projektu umieścę tę właśnie notatkę i użyję komendy **'git add .'** aby dodać wszystkie (w tym przypadku ten jeden plik) pliki z aktualnego katalogu, które nie są jeszcze śledzone.

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add .
```

```
$ git commit
```

Po dodaniu zmian z katalogu, Git jeszcze faktycznie ich nie śledzi, aby to osiągnąć, należy wywołać komendę **git commit** z dodatkową flagą **-m "i naszym komentarzem"**.

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git commit -m "[Initial Commit] Added my learning notes"
[master (root-commit) 7dfde50] [Initial Commit] Added my learning notes
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 DevOps_Notes.pdf
```

```
git commit -- amend
```

W przypadku gdybyśmy chcieli dodać nowe zmiany do **istniejącego** commit'a, skorzystamy komendy **git commit --amend -m "nasz komentarz"**.

Ta opcja działa tylko dla **ostatniego** commitu w naszej hierarchii. Nie możemy dodać nowych zmian do innego, wcześniejszego, commitu.

```

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git show 074168e922d5de01ce2a8b4372c18a57075590ec
commit 074168e922d5de01ce2a8b4372c18a57075590ec (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 15:25:11 2022 +0200

    Adds file to commit    ZAWARTOŚĆ POPRZEDNIEGO COMMITA

diff --git a/explore_california/file_to_commit.txt b/explore_california/file_to_commit.txt
new file mode 100644
index 0000000..e69de29

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 12 commits.
  (use "git push" to publish your local commits)

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

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

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git add file_to_amend.txt

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git commit --amend -m "Add file_to_commit & file_to_amend"
[master 23c3318] Add file_to_commit & file_to_amend
Date: Sat Oct 1 15:25:11 2022 +0200
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 explore_california/file_to_amend.txt
create mode 100644 explore_california/file_to_commit.txt

```

```

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git show 23c331891ce55ba769409b96ba2cd47845c4b4b0
commit 23c331891ce55ba769409b96ba2cd47845c4b4b0 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 15:25:11 2022 +0200

    Add file_to_commit & file_to_amend

diff --git a/explore_california/file_to_amend.txt b/explore_california/file_to_amend.txt
new file mode 100644
index 0000000..e69de29    STARY COMMIT ZOSTAŁ ZASTĄPIONY NOWYM Z NOWYMI ZMIANAMI
diff --git a/explore_california/file_to_commit.txt b/explore_california/file_to_commit.txt
new file mode 100644
index 0000000..e69de29

```

Wynikiem operacji jest zupełnie nowy commit (z innym **SHA**) zawierający wewnątrz poprzedniego + nowo dodane zmiany.

```
$ git log
```

Komenda **git log** wylistuje wszystkie commit'y i informację o nich, które zostały utworzone w ramach naszego projektu.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git log
commit 7dfde5067f9e1c7d246e9eec1d35791054e238c3 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date:   Wed Sep 7 00:34:20 2022 +0200

    [Initial Commit] Added my learning notes
```

Idąc od góry, widzimy:

- commit** - Unikalny numer identyfikujący nasz commit.
- Author** - Autor commit, pobrany z pliku konfiguracyjnego `.gitconfig`.
- Date** - Data utworzenia commit.
- message** - Komentarz dodany przez autora w trakcie tworzenia commit komendą `git commit -m "komentarz"`

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git log --since=2022-09-01 --until=2022-09-31 --author=mikosz --grep="Init"
commit 7dfde5067f9e1c7d246e9eec1d35791054e238c3
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date:   Wed Sep 7 00:34:20 2022 +0200

    [Initial Commit] Added my learning notes
```

```
$ git status
```

Zadaniem komendy **git status** jest wylistowanie wykrytych zmian wewnątrz projektu.

Przykładowe przypadki użycia:

Brak zmian:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Wykrycie nowego pliku:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    new_file.txt.txt

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

Zarejestrowanie śledzenia nowego pliku:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add new_file.txt.txt

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   new_file.txt.txt
```

Zarejestrowanie usunięcia nowego pliku:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   new_file.txt.txt

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

```
$ git diff
```

Podczas wprowadzania zmian w wielu plikach (już istniejących w naszym repozytorium), czasem możemy zapomnieć, lub nawet nie spodziewać się, że jakiś plik zawiera w sobie nowe elementy. W takich chwilach dobrze jest posłużyć się komendą **git diff**, która wylistuje nam istniejące modyfikacje.

W tym przykładzie dodałem do repozytorium plik **file.txt**, po czym go zmodyfikowałem:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   file.txt

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:   file.txt
```

Komendą **git diff** mogę podejrzeć co zostało zmienione:

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git diff
diff --git a/file.txt b/file.txt
index f925f18..0cd5375 100644
--- a/file.txt
+++ b/file.txt
@@ -1,1 @@
-Hello, this is a file.
\ No newline at end of file
+Hello, this is a modified file.
\ No newline at end of file
```

Jak widać kolorem **czernym** został zaznaczony fragment, który uległ zmianie i kolorem **niebieskim** jak wygląda teraz.

Domyślnie **git diff** pokaże nam zmiany istniejące pomiędzy **working directory** a **staging area** (więcej o nich później), czyli pomiędzy naszym lokalnym miejscem roboczym a obszarem, w którym git przechowuje poprzednią wersję pliku. Czyli po użyciu komendy **git add file.txt** nie zobaczymy już zmian.

W przypadku gdybyśmy chcieli zobaczyć zmiany na już dodanych plikach skorzystamy z komendy **git diff --staged**.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add file.txt

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git diff --staged
diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..3aae3e9
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+Hello, this is a modified and staged file.
\ No newline at end of file
```

W tym przypadku ujrzymy zmiany pomiędzy *repository* a *staging area* (tak jakby poziom wyżej).

```
$ git rm --cached
```

W przypadku gdy dodamy do **staging area** niechciany plik, możemy go usunąć przy pomocy **git rm --cached <nazwa_pliku>**. Ta operacja usunie go z **indexu** ale sam plik **pozostanie** w folderze roboczym.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Your branch is up to date with 'DevOps/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   file_to_delete.txt

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git rm --cached file_to_delete.txt
rm 'file_to_delete.txt'

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Your branch is up to date with 'DevOps/master'.

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

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

```
$ git show
```

Komenda **git show <sha>** powie nam co zawiera dany **commit**. (polecam flagę - **-color-words**)

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git log --since 2022.09.30
commit 71a0b08b9862dcf641f40cde92e8467fcb870bdc (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 11:50:23 2022 +0200

    Changes 'you'll' to 'you will'

commit 826e1699c3f1471215b241dbbc47da34928e1f66
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 11:45:40 2022 +0200

    Edits support phone number

commit f932b7dabd350081648df65af993174a42ac8d73
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 11:33:56 2022 +0200

    Adds training project

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git show 71a0b08b9862dcf641f40cde92e8467fcb870bdc
commit 71a0b08b9862dcf641f40cde92e8467fcb870bdc (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 11:50:23 2022 +0200

    Changes 'you'll' to 'you will'

diff --git a/explore_california/index.html b/explore_california/index.html
index 23f0d6d..84abca3 100644
--- a/explore_california/index.html
+++ b/explore_california/index.html
@@ -61,7 +61,7 @@
     <div id="mainContent">
       <div id="mainArticle">
         <h1>Tour Spotlight</h1>
-        <p class="spotlight">This month's spotlight package is Cycle Cal
+        <p class="spotlight">This month's spotlight package is Cycle Cal
 ifornia. Whether you are looking for some serious downhill thrills to a relax
 ing ride along the coast, you'll find something to love in Cycle California.<
 br /> <span class="accent"><a href="tours_cycle.html" title="Cycle California
">tour details</a></span></p>
+        <p class="spotlight">This month's spotlight package is Cycle Cal
 ifornia. Whether you are looking for some serious downhill thrills to a relax
 ing ride along the coast, you will find something to love in Cycle California
 .<br /> <span class="accent"><a href="tours_cycle.html" title="Cycle Californ
 ia">tour details</a></span></p>

         <h1>Explorer's Podcast</h1>
         <p class="videoText">Join us each month as we publish a new Expl
 ore California video podcast, with featured tours, customer photos, and some
 exciting new features!<span class="accent"><a href="explorers/video.html" tit
 le="Video Podcast">Watch the full video</a></span></p>

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
```

```
$ git checkout
```

`git reset` changes the staging index and `git checkout` changes the working directory.

```
$ git checkout -- <file_name>
```

Załóżmy, że wyłączyłem edytor i okazało się, że przez przypadek usunąłem kawałek istotnego kodu:

```
ziolle@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git diff
diff --git a/explore_california/index.html b/explore_california/index.html
index d39b10e..c2a648d 100644
--- a/explore_california/index.html
+++ b/explore_california/index.html
@@ -57,16 +57,7 @@
     <a href="tours.html" title="Find your tour!"><h2>Find your tour</h2></a>
   </div>

   <div id="contentwrapper">
     <div id="mainContent">
       <div id="mainArticle">
         <h1>Tour Spotlight</h1>
         <p class="spotlight">This month's spotlight package is Cycle California. Whether you
are looking for some serious downhill thrills to a relaxing ride along the coast, you will find
something to love in Cycle California.<br /> <span class="accent"><a href="tours_cycle.html" title="Cycle California">tour details</a></span></p>
       </div>
       <div id="videoText">Join us each month as we publish a new Explore California video
podcast, with featured tours, customer photos, and some exciting new features!<span class="accent"><a href="explorers/video.html" title="Video Podcast">watch the full video</a></span></p>
       </div>
     </div>

     <div id="secondaryContent">
       <div id="specials" class="callout">

```

Nie mam już możliwości cofnięcia zmian z **ctr + z**. Na szczęście Git śledził poprzednie wersje tego pliku i teraz mogę w łatwy sposób polecić mu sprawdzenie danego pliku i **przywrócenie aktualnej wersji z repozytorium**, w tym przypadku użyję opcji **git checkout -- index.html**.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git checkout -- index.html

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 10 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
$git checkout <SHA> -- <file_name>
```

W przypadku jeśli chcemy przywrócić wersję pliku z konkretnego commitu, do komendy **git checkout** musimy jeszcze dorzucić **SHA** i nazwę pliku, którego wersję chcemy przywrócić.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git show dfa1107d3f2c70a5f91729f9a22bc9ac8c91b0c8
commit dfa1107d3f2c70a5f91729f9a22bc9ac8c91b0c8 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 15:48:19 2022 +0200

    Edits some_file

diff --git a/explore_california/some_file.txt b/explore_california/some_file.txt
index f158bc7..af9f7d0 100644
--- a/explore_california/some_file.txt
+++ b/explore_california/some_file.txt
@@ -1,1 @@
-This is old message
+This is new message
```

Załóżmy, że chcemy przywrócić starą wersję pliku **some_file.txt**. Aktualnie w jego zawartości znajdziemy wiadomość "This is new message".

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git checkout 3bd693067110eb94158f15518f291f23525d6a41 -- some_file.txt

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 14 commits.
  (use "git push" to publish your local commits)

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

Po wprowadzeniu, wcześniej znalezionej (np.: komendą **git log**) SHA, Git automatycznie przywrócił poprzednią wersję pliku i umieścił ją w **staging area**.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git diff --staged
diff --git a/explore_california/some_file.txt b/explore_california/some_file.txt
index af9f7d0..f158bc7 100644
--- a/explore_california/some_file.txt
+++ b/explore_california/some_file.txt
@@ -1,1 @@
-This is new message
+This is old message
```

```
$ git reset
```

Resetowanie przyda się w momencie gdy umieściliśmy już zmiany w **staging area**, tzn zrobiliśmy **git add .**, i chcielibyśmy cofnąć ten proces.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 10 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   tours.html
    modified:   tours/tour_detail_backpack_cal.html
    modified:   tours/tour_detail_bigsur.html

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:   contact.html
    modified:   explorers.html
    modified:   index.html
    modified:   mission.html
    modified:   resources.html
    modified:   resources/faq.html
    modified:   support.html
```

Załóżmy, że jednak **tours.html** nie powinien się znaleźć w **staging area**, przywrócimy go za pomocą komendy **git reset HEAD tours.html** :

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git reset HEAD tours.html
Unstaged changes after reset:
M   explore_california/contact.html
M   explore_california/explorers.html
M   explore_california/index.html
M   explore_california/mission.html
M   explore_california/resources.html
M   explore_california/resources/faq.html
M   explore_california/support.html
M   explore_california/tours.html

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/explore_california (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 10 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   tours/tour_detail_backpack_cal.html
    modified:   tours/tour_detail_bigsur.html

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:   contact.html
    modified:   explorers.html
    modified:   index.html
    modified:   mission.html
    modified:   resources.html
    modified:   resources/faq.html
    modified:   support.html
    modified:   tours.html
```

Ten zabieg nazywamy **unstaging**iem.

```
$ git revert
```

W przypadku gdy, na przykład, okazało się, że nowy **commit** zawiera zmiany, które są całkowicie błędne i niepotrzebne, chcielibyśmy je cofnąć.

```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git show 1960ec332ed18fb2b29ed20ff3954ba9eec29cff
commit 1960ec332ed18fb2b29ed20ff3954ba9eec29cff (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 17:09:10 2022 +0200

    Adds working file

diff --git a/working_file.txt b/working_file.txt
new file mode 100644
index 0000000..68300b8
--- /dev/null
+++ b/working_file.txt
@@ -0,0 +1 @@
+It works!

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git show a64d005661bf95855d6b3b5ab0b013fb4d65b378
commit a64d005661bf95855d6b3b5ab0b013fb4d65b378 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 17:10:11 2022 +0200

    Break file

diff --git a/working_file.txt b/working_file.txt
index 68300b8..b309c62 100644
--- a/working_file.txt
+++ b/working_file.txt
@@ -1,1 +1,1 @@
-It works!
+It does not work! :(
```

Możemy to osiągnąć komendą **git revert <SHA>**. Gdzie SHA pochodzi od commitu, który spowodował niechciane zmiany.

```

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git revert a64d005661bf95855d6b3b5ab0b013fb4d65b378
[master b4cf76f] Revert "Break file"
1 file changed, 1 insertion(+), 1 deletion(-)

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git show b4cf76ffb26bf963134b368088b5fc65e9ab8ac9
commit b4cf76ffb26bf963134b368088b5fc65e9ab8ac9 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Sat Oct 1 17:11:19 2022 +0200

    Revert "Break file"

    This reverts commit a64d005661bf95855d6b3b5ab0b013fb4d65b378.

diff --git a/working_file.txt b/working_file.txt
index b309c62..68300b8 100644
--- a/working_file.txt
+++ b/working_file.txt
@@ -1,1 @@
-It does not work! :(
+It works!

```

```
$ git clean
```

Sprzątanie plików-śmieci, takich jak .log, .err, może być uciążliwe, zwłaszcza jak liczymy je w dziesiątkach lub setkach.

Aby wyczyścić naszą stację roboczą z nie-śledzonych plików (**untracked files**) użyjemy komendy **git clean -f**.

Dodatkowym krokiem bezpieczeństwa będzie użycie flagi **-n**, która pokaże nam co by się stało gdybyśmy rzeczywiście uruchomili ten skrypt. (Zalecane podejście ponieważ usuwanie plików skryptem często jest niebezpieczne).

```

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Your branch is ahead of 'DevOps/master' by 35 commits.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        junk_01.txt
        junk_02.txt
        junk_03.txt

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

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git clean -n
Would remove junk_01.txt
Would remove junk_02.txt
Would remove junk_03.txt

```

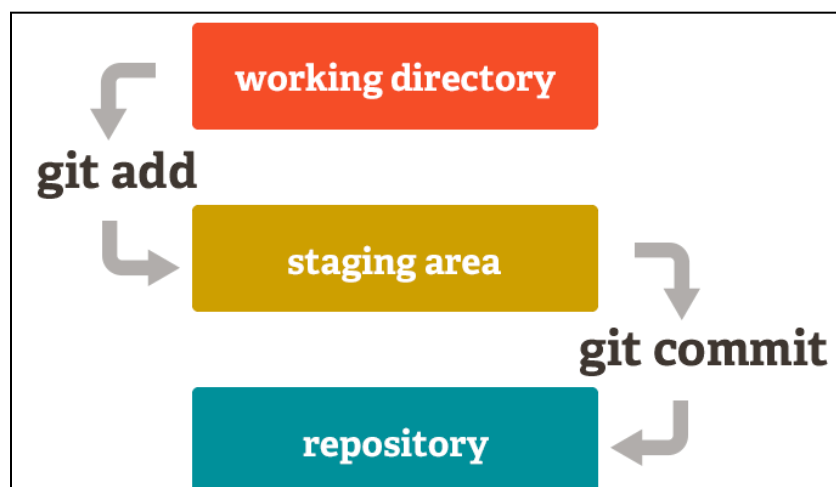
W momencie gdy jesteśmy pewni, że pliki, które mają zostać usunięte są właściwe, uruchamiamy komendę z flagą -f, force.

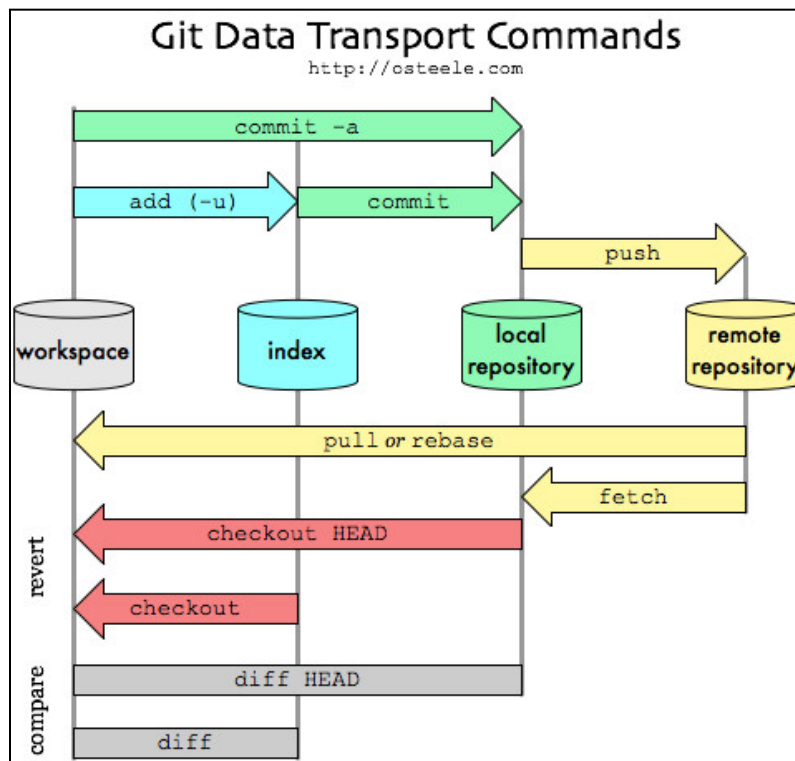
```
ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git clean -f
Removing junk_01.txt
Removing junk_02.txt
Removing junk_03.txt
```

Git Architecture

Git używa tak zwanej **architektury trzech drzew (three-tree architecture)**.

To oznacza, że (tak jak inne systemy kontroli) posiada repozytorium i strefę roboczą i dodatkowo **staging index** czyli miejsce, do którego odkładane są nasze **commit'y**.





Dzięki temu możemy dokonać zmian w wielu różnych plikach w naszym obszarze roboczym, a następnie utworzyć zestaw z kilku niezależnych commitów.

Hash Values & HEAD

Hash Value

Git generuje sumę kontrolną (checksum) dla każdego zbioru zmian.

Jest to wartość **HASH** - liczba generowana przez pobranie danych i wprowadzenie ich do algorytmu matematycznego.

W folderze `.git` wewnątrz katalogu naszego projektu znajdziemy plik **'HEAD'** - znajdziemy w nim informację o tym na jaką wartość hash, w tym momencie, wskazuje HEAD.

Przy pomocy komendy `'git log'` możemy sprawdzić czy faktycznie tak jest.

```

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ cat HEAD
ref: refs/heads/master

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ cat refs/heads/master
e4d974efe63ed09495b02a00a68643c8536f0d76

ziol@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ git log
commit e4d974efe63ed09495b02a00a68643c8536f0d76 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date: Thu Sep 8 02:23:55 2022 +0200

    Adds new topic - 'Git Architecture'

```

Jak widać wartość w pliku HEAD wskazuje na hash zawarty w moim ostatnim commit'cie.

H-E-A-D

Jest to zmienna nazywana wskaznikiem lub referencją do konkretnego **commit'a**.

HEAD zawsze wskazuje na 'czubek' naszego branch'a (o branch'ach później) w naszym repozytorium. To miejsce w którym zakończyliśmy nasze ostatnie działania.

