

Systemy kontroli wersji

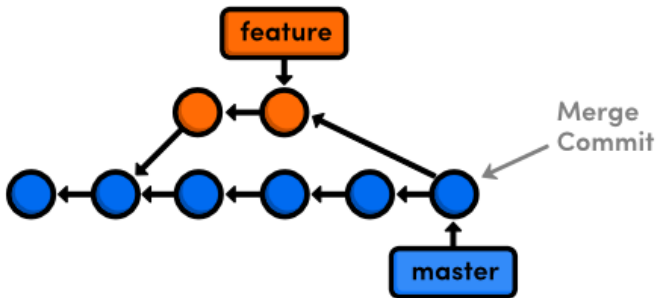
Git - cykl pracy

Adam Kobus[†]

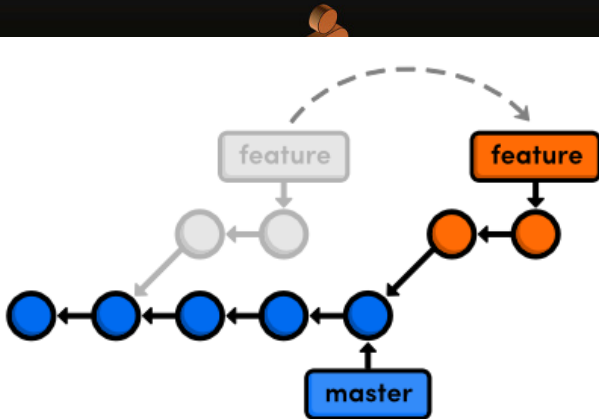
12 listopada 2020

[†]adam.kobus@poczta.umcs.lublin.pl

Merge

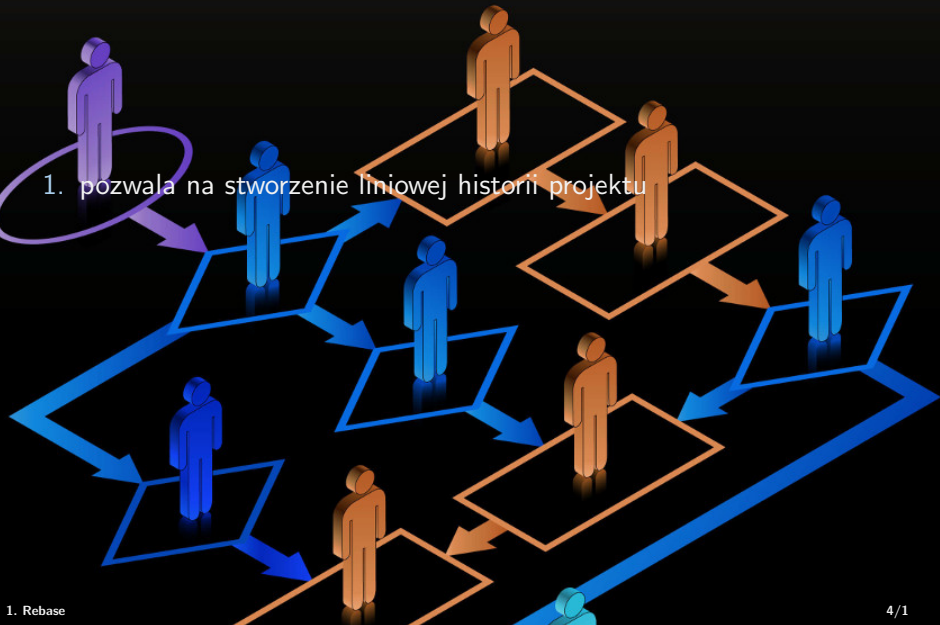


Rebase

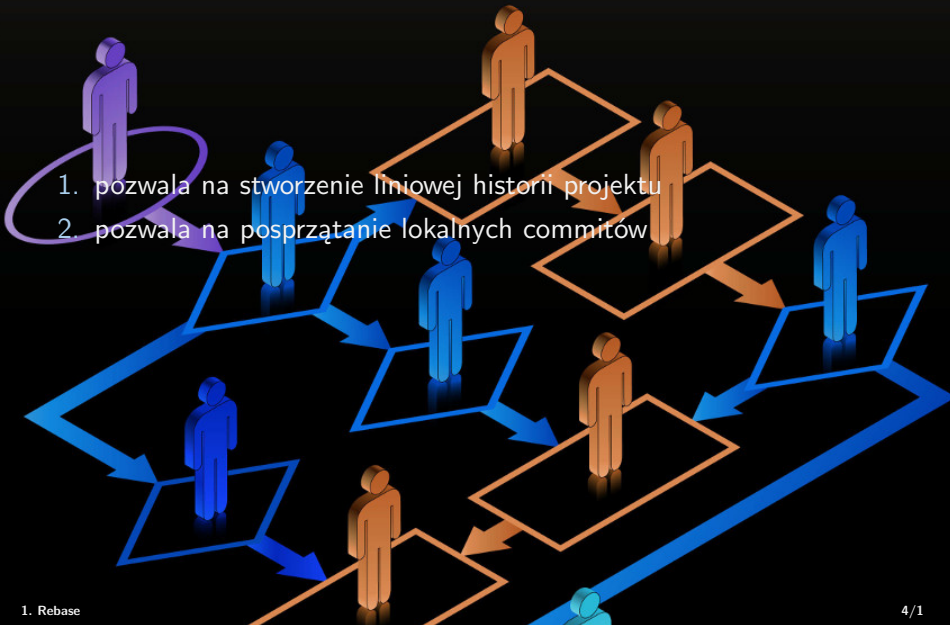


Rebase

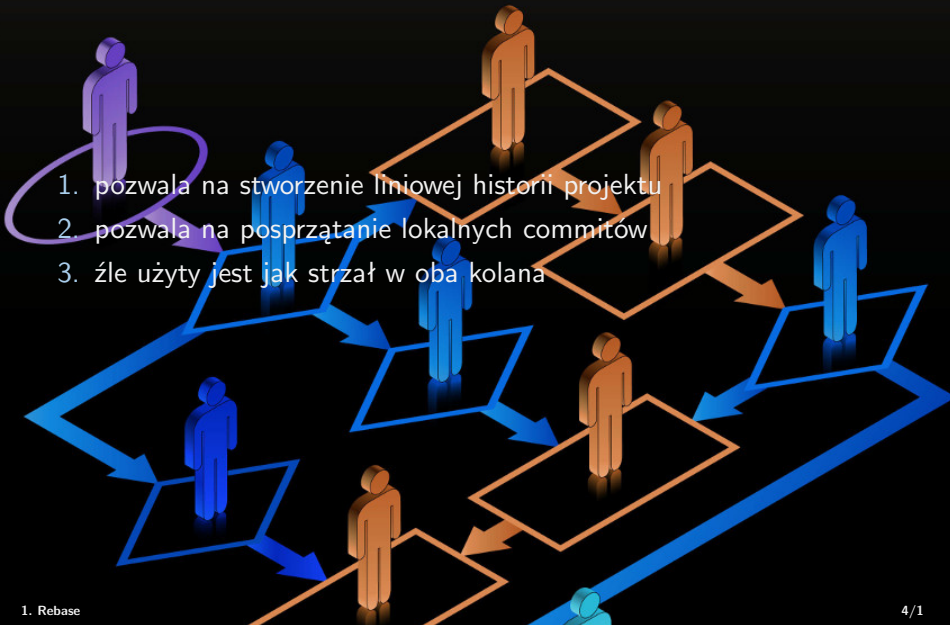
1. pozwala na stworzenie liniowej historii projektu

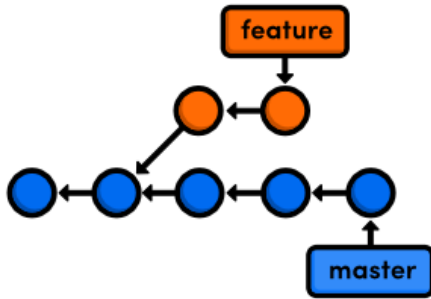


Rebase

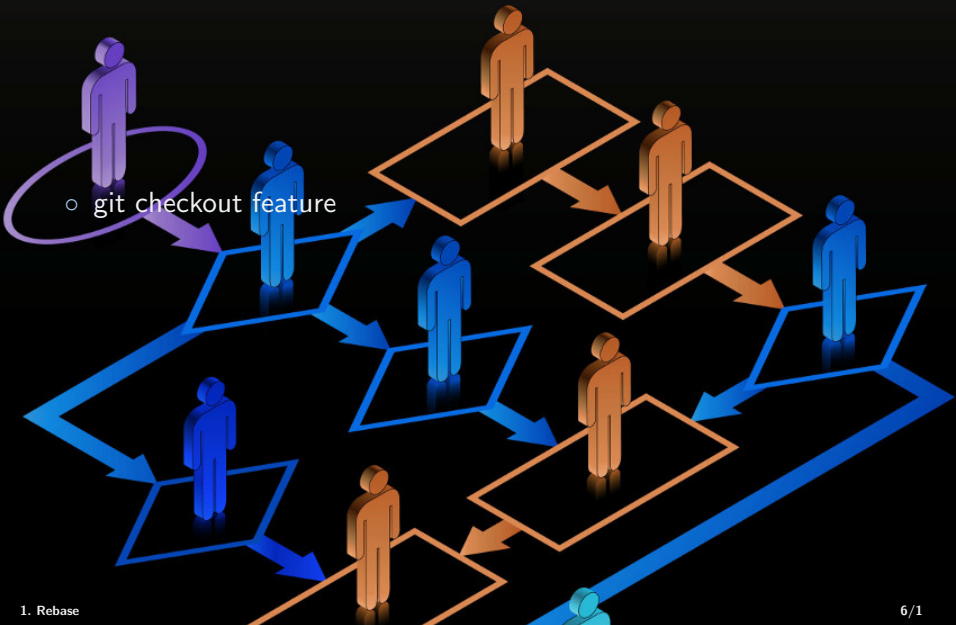
- 
1. pozwala na stworzenie liniowej historii projektu
 2. pozwala na posprzątanie lokalnych commitów

Rebase

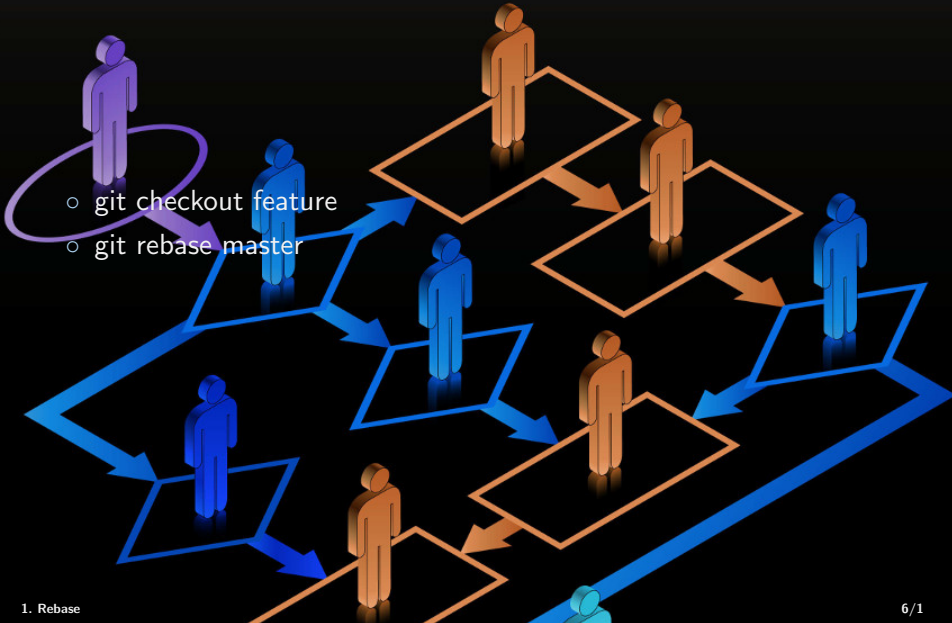
- 
1. pozwala na stworzenie liniowej historii projektu
 2. pozwala na posprzątanie lokalnych commitów
 3. źle użyty jest jak strzał w oba kolana

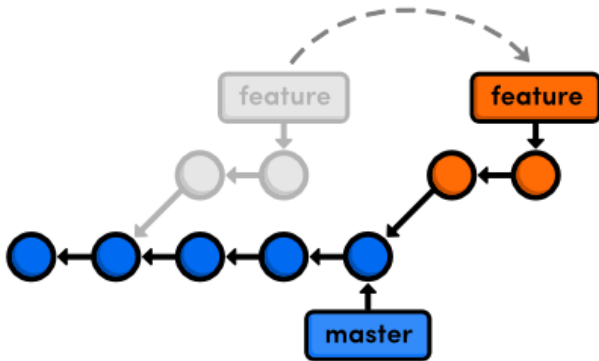


Rebase



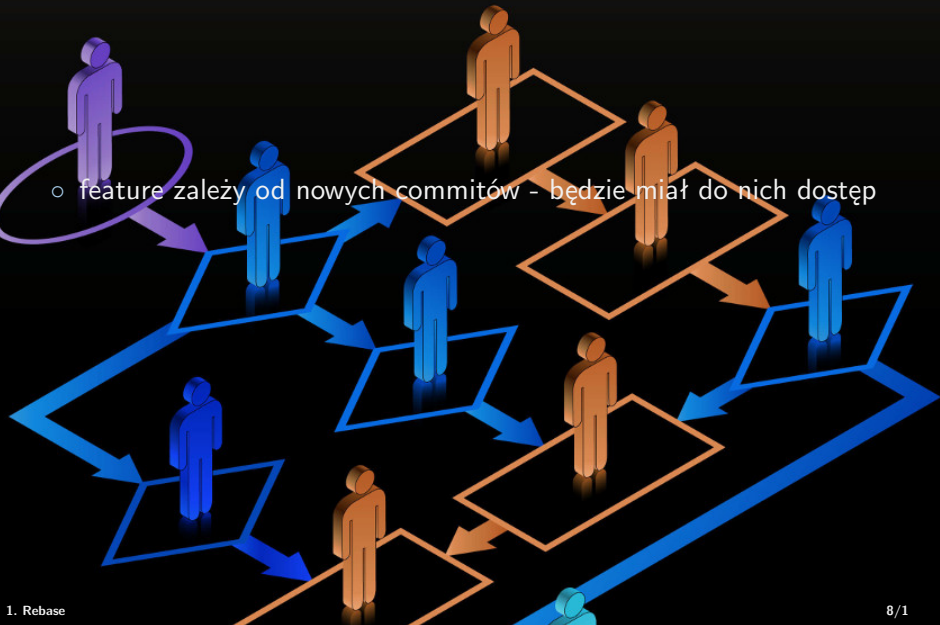
Rebase





Scenariusze do rebase

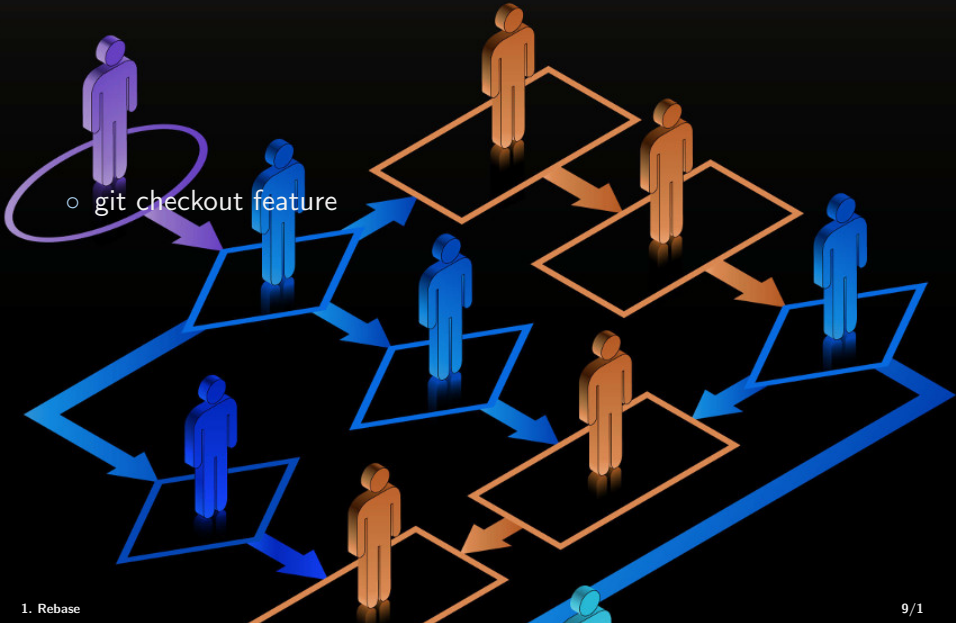
- feature zależy od nowych commitów - będzie miał do nich dostęp



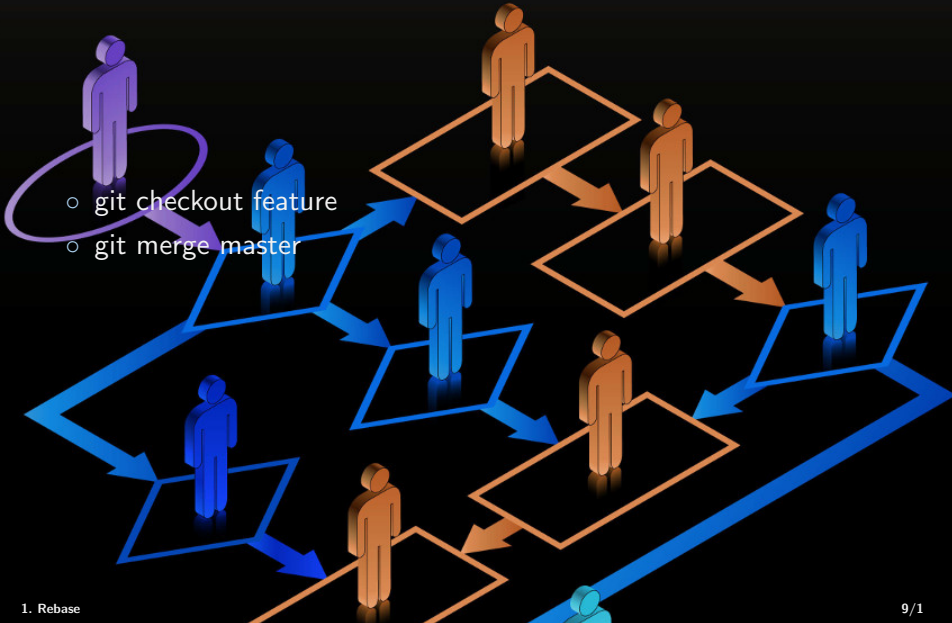
Scenariusze do rebase

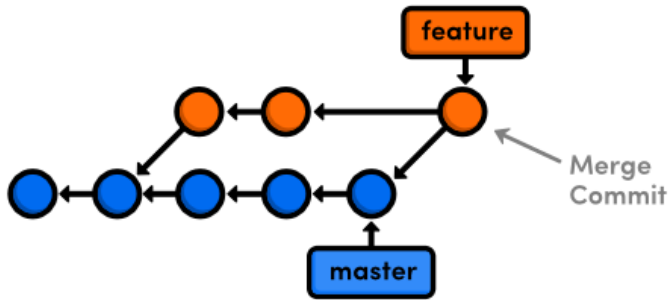
- feature zależy od nowych commitów - będzie miał do nich dostęp
- feature jest zakończony - master może przesunąć się poprzez fast forward

Rebase

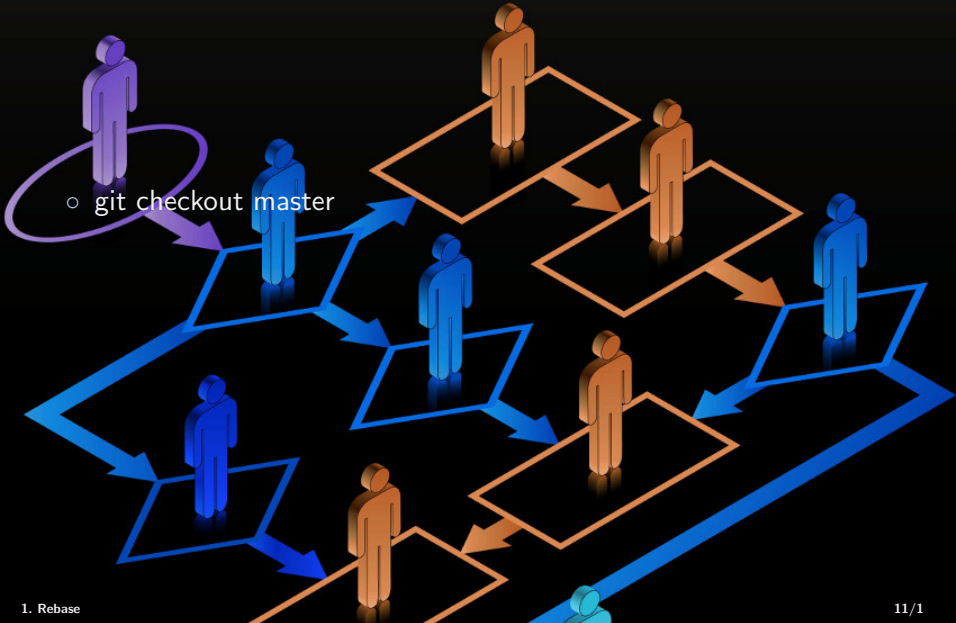


Rebase

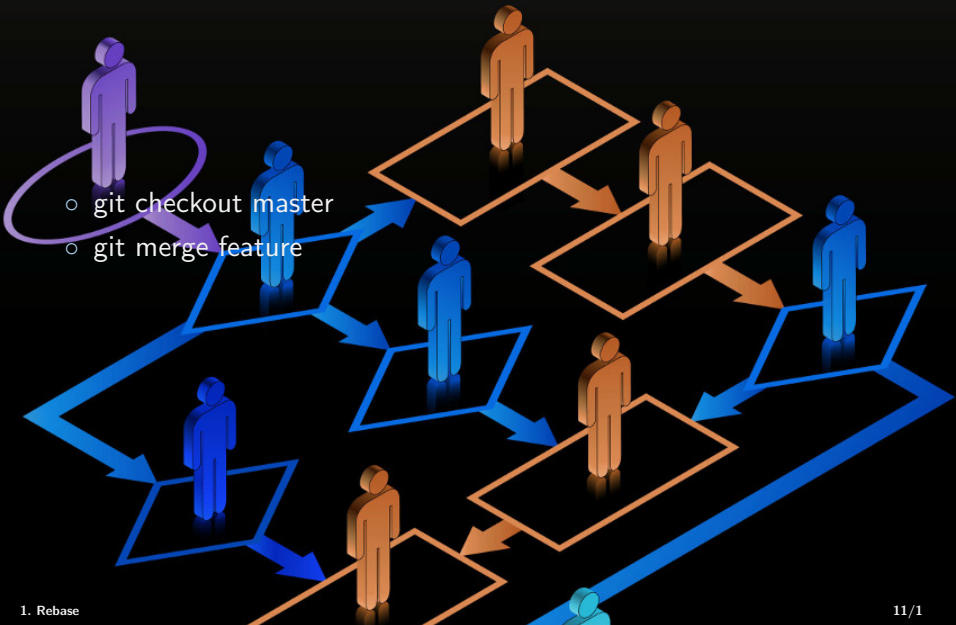


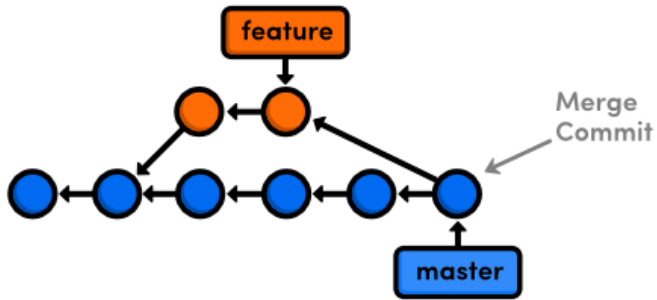


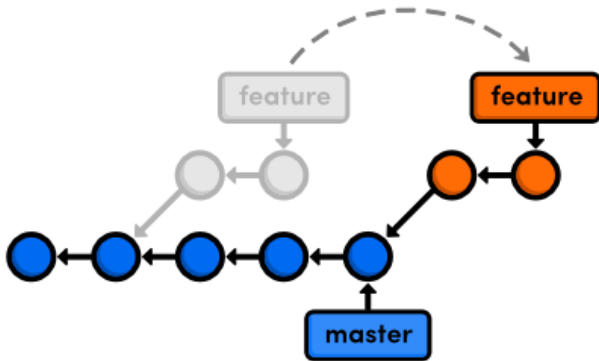
Rebase

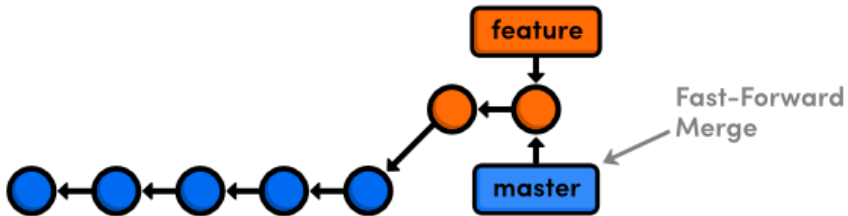


Rebase

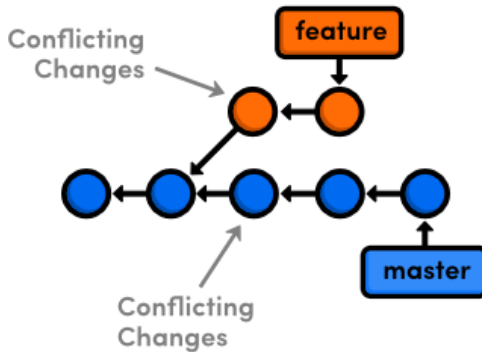








Konflikty przy rebase



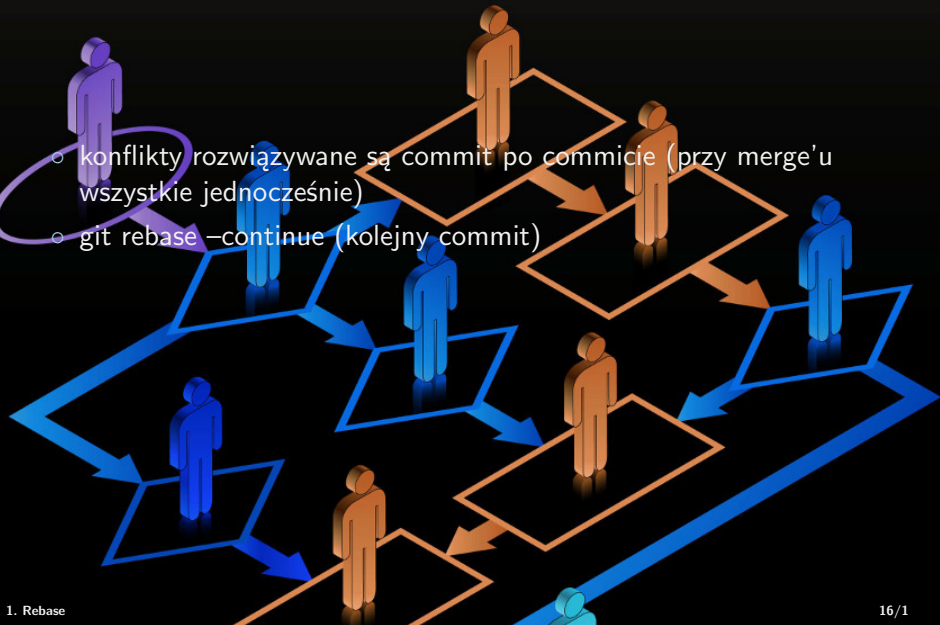
Rozwiązywanie konfliktów przy rebase

- konflikty rozwiązywane są commit po commit (przy merge'u wszystkie jednocześnie)



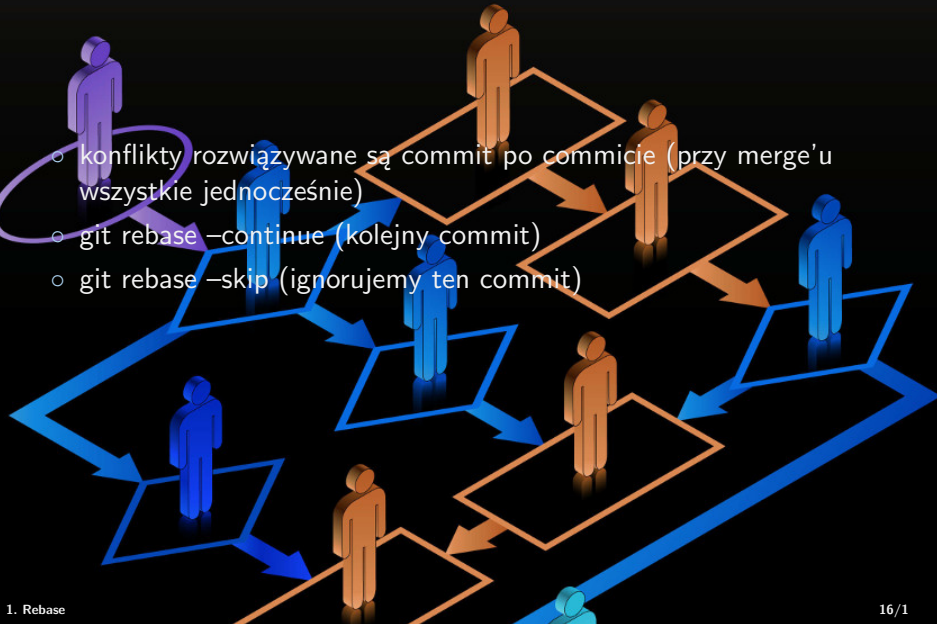
Rozwiązywanie konfliktów przy rebase

- o konflikty rozwiązywane są commit po commitcie (przy merge'u wszystkie jednocześnie)
- o git rebase -continue (kolejny commit)

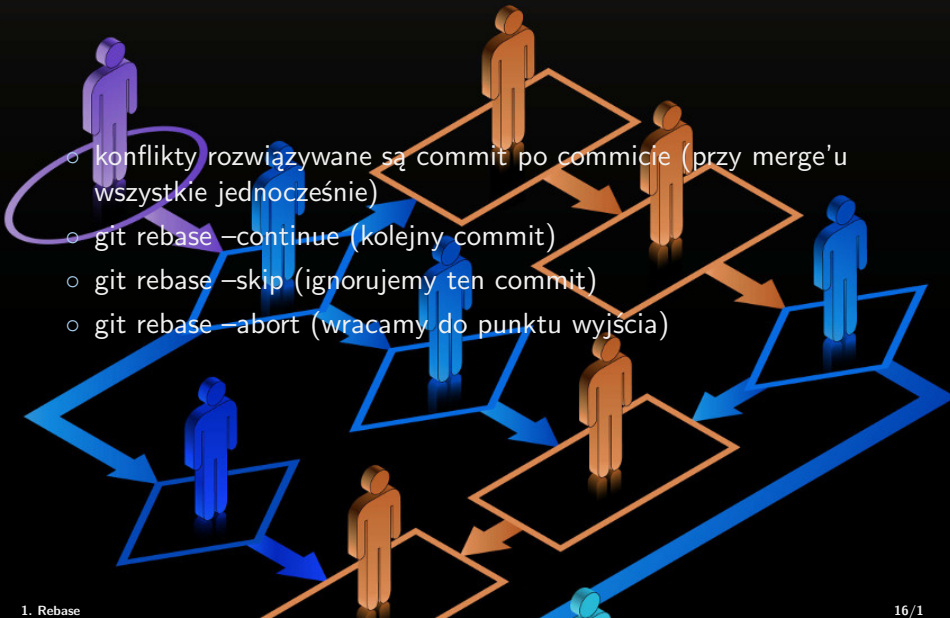


Rozwiązywanie konfliktów przy rebase

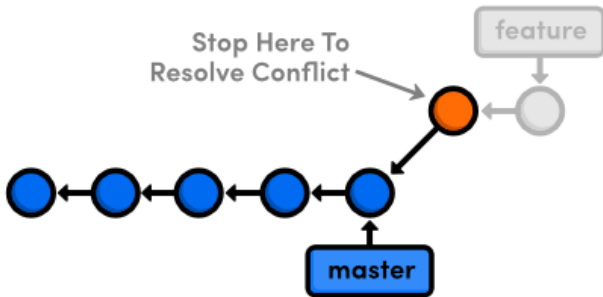
- konflikty rozwiązywane są commit po commit (przy merge'u wszystkie jednocześnie)
- git rebase -continue (kolejny commit)
- git rebase -skip (ignorujemy ten commit)



Rozwiązywanie konfliktów przy rebase

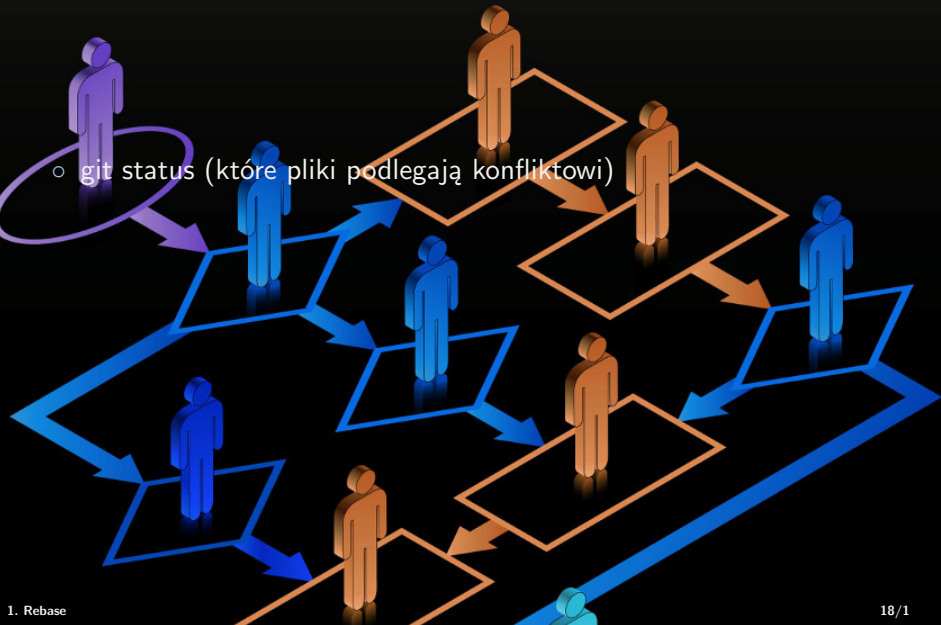
- 
- o konflikty rozwiązywane są commit po commitcie (przy merge'u wszystkie jednocześnie)
 - o `git rebase --continue` (kolejny commit)
 - o `git rebase --skip` (ignorujemy ten commit)
 - o `git rebase --abort` (wracamy do punktu wyjścia)

Konflikty przy rebase



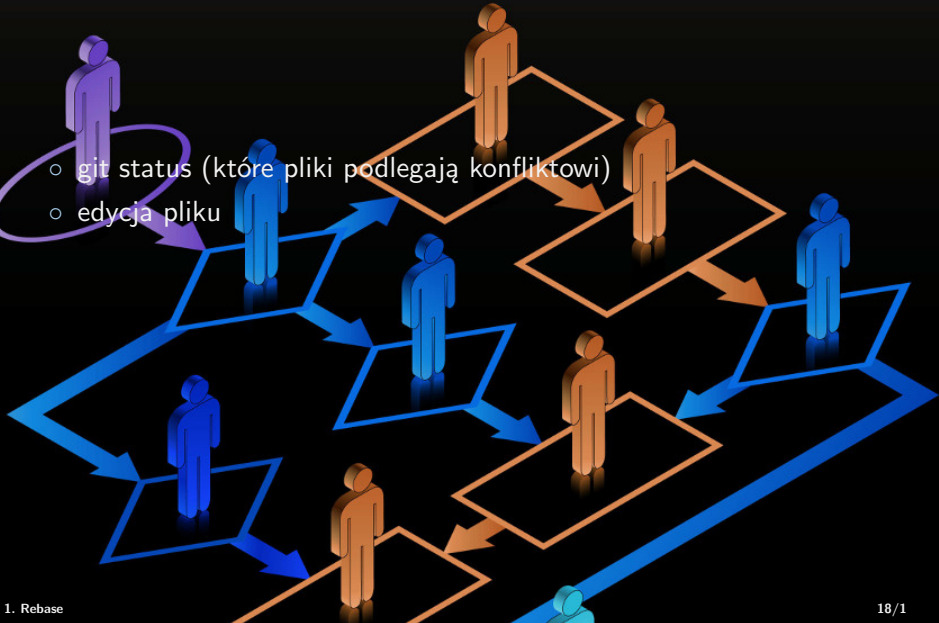
Rozwiązywanie konfliktu przy rebase

- git status (które pliki podlegają konfliktowi)



Rozwiązywanie konfliktu przy rebase

- git status (które pliki podlegają konfliktowi)
- edycja pliku



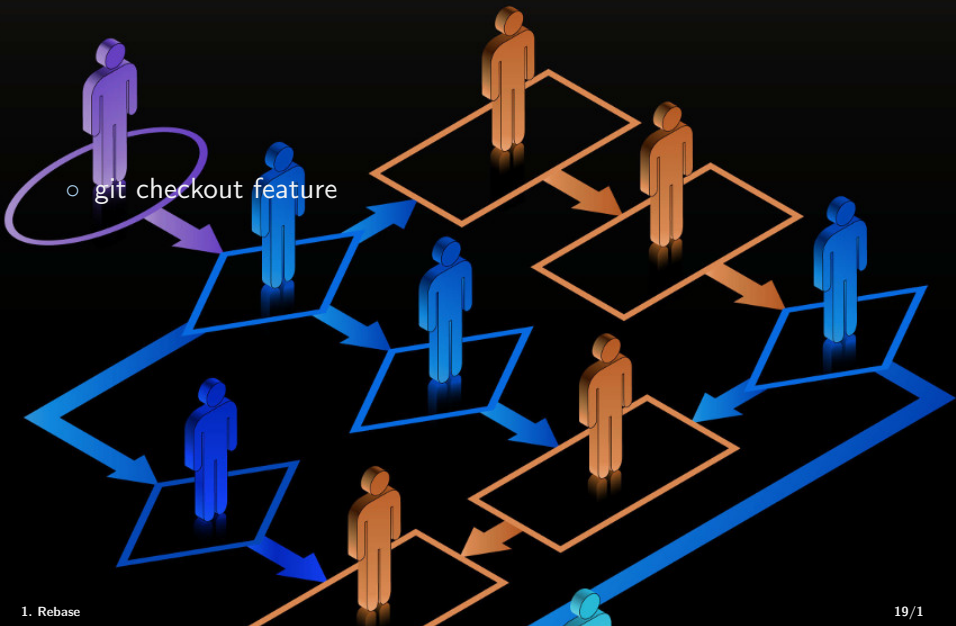
Rozwiązywanie konfliktu przy rebase

- 
- git status (które pliki podlegają konfliktowi)
 - edycja pliku
 - git add plik.txt (rozwiązanie konfliktu)

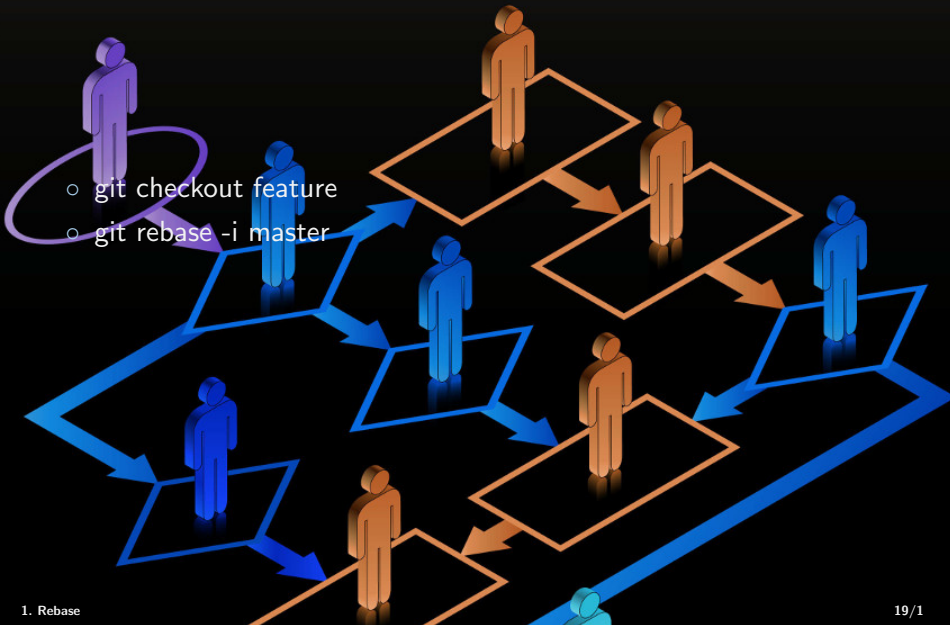
Rozwiązywanie konfliktu przy rebase

- 
- git status (które pliki podlegają konfliktowi)
 - edycja pliku
 - git add plik.txt (rozwiązanie konfliktu)
 - git rebase -continue (kolejny commit)

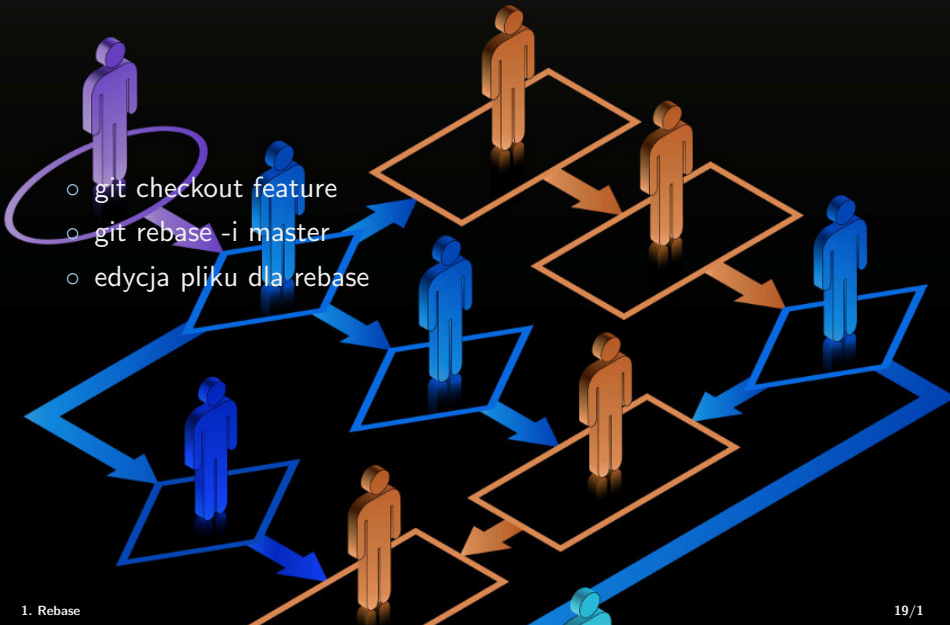
Interaktywny rebase



Interaktywny rebase

- git checkout feature
 - git rebase -i master
- 
- The diagram illustrates an interactive rebase workflow. It shows a sequence of commits represented by human figures on a grid. A purple circle highlights the first commit, with arrows pointing to the second and third commits. The workflow involves checking out a feature branch and then rebasing master onto it.

Interaktywny rebase

- git checkout feature
 - git rebase -i master
 - edycja pliku dla rebase
- 

Interaktywny rebase

```
adam@falco:/workspace/remotealpha/alpha$ git checkout feature
Switched to branch 'feature'
adam@falco:/workspace/remotealpha/alpha$ touch testrebase.txt
adam@falco:/workspace/remotealpha/alpha$ git add testrebase.txt
adam@falco:/workspace/remotealpha/alpha$ git commit -m "Test rebase"
[feature 241305c] Test rebase
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testrebase.txt
adam@falco:/workspace/remotealpha/alpha$ touch testrebase2.txt
adam@falco:/workspace/remotealpha/alpha$ git add testrebase2.txt
adam@falco:/workspace/remotealpha/alpha$ git commit -m "Test rebase 2"
[feature 6378f2a] Test rebase 2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testrebase2.txt
adam@falco:/workspace/remotealpha/alpha$ git log --oneline --graph --decorate
* 6378f2a (HEAD, feature) Test rebase 2
* 241305c Test rebase
* f29e6b3 (tag: v0.1alfa, tag: v0.1, origin/master, origin/develop, master, develop) feature1 merged to
\
| * ab50a92 (origin/feature1, feature1) feature1 Element
* | 6e0d2f3 develop Element
|/
* 783da15 (tag: v0.09) Elements
* 79b4a33 Conflict resolved with meld
\
| * 301268d feature1 - deleted and modified line
* | 6f59d6a develop - changed line
|/
* da6e56b Merge branch 'feature1' into feature2
\
| * b9429de Another line from feature1
* | eb81425 Another line from feature2
|/
```

Konflikty przy rebase

```
adam@falco:/workspace/remotealpha/alpha$ git rebase -i master
```

Interaktywny rebase

```
GNU nano 2.2.6 Plik: /workspace/remotearpha/alpha/.git/rebase-merge/git-rebase-todo

pick 241305c Test rebase
pick 6378f2a Test rebase 2

# Rebase f29e6b3..6378f2a onto f29e6b3
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out

[ Wczytano 20 linii ]
^G Pomoc      ^O Zapisz      ^R Wczyt.plik  ^Y Poprz.str.  ^K Wytnij      ^C Bież.poz.
^X Wyjdź      ^J Wyjustuj   ^W Wyszukaj   ^V Nast.str.   ^U Wklej       ^T Pisownia
```

Konflikty przy rebase

```
GNU nano 2.2.6 Plik: /workspace/remotealpha/alpha/.git/rebase-merge/git-rebase-todo Zmodyfikowany
pick 6378f2a Test rebase 2

# Rebase f29e6b3..6378f2a onto f29e6b3
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out

^G Pomoc      ^O Zapisz      ^R Wczyt.plik  ^Y Poprz.str.  ^K Wytnij     ^C Bież.poz.
^X Wyjdź      ^J Wyjustuj   ^W Wyszukaj   ^V Nast.str.  ^U Wklej      ^T Pisownia
```

Interaktywny rebase

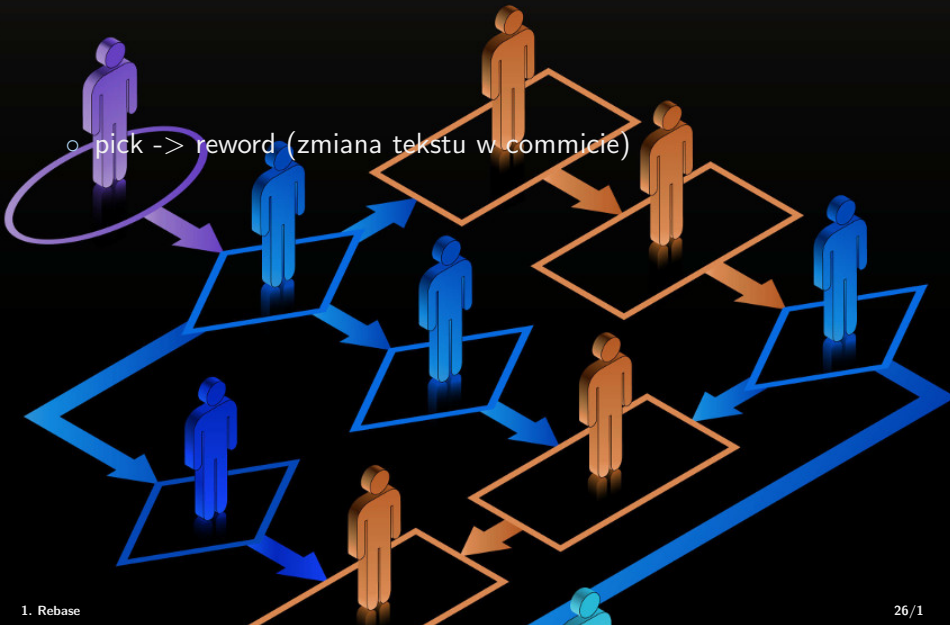
```
adam@falco:/workspace/remotealpha/alpha$ git rebase -i master  
Successfully rebased and updated refs/heads/feature.  
adam@falco:/workspace/remotealpha/alpha$
```

Konflikty przy rebase

```
adam@falco:/workspace/remotealpha/alpha$ git rebase -i master
Successfully rebased and updated refs/heads/feature.
adam@falco:/workspace/remotealpha/alpha$ ls
Element.class  Element.java  Element.java~  Hello.class  Hello.java  Hello.java~  testrebase2.txt
adam@falco:/workspace/remotealpha/alpha$
```

Interaktywny rebase

- pick -> reword (zmiana tekstu w commicie)



Interaktywny rebase

- pick -> reword (zmiana tekstu w commicie)
- pick -> squash (złączenie dwóch commitów w jeden, na bazie wcześniejszego z listy), po wyjściu z edytora, procedura zgodnie z plikiem będzie zatrzymywała się, by można było dokonać 'git commit -amend'

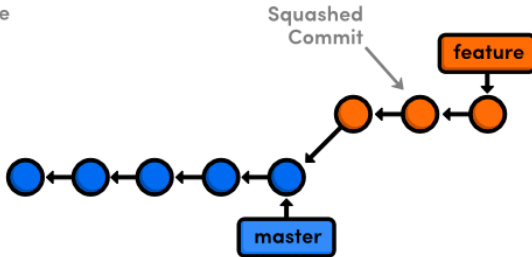


Interaktywny rebase

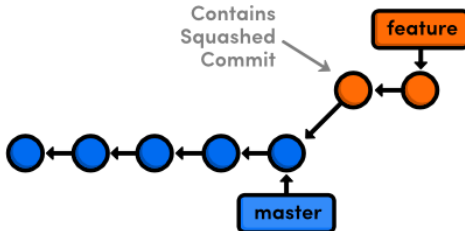
- pick -> reword (zmiana tekstu w commicie)
- pick -> squash (złączenie dwóch commitów w jeden, na bazie wcześniejszego z listy), po wyjściu z edytora, procedura zgodnie z plikiem będzie zatrzymywała się, by można było dokonać 'git commit --amend'
- można również pomiędzy nie dodać commity (git add, git commit)

Interaktywny rebase

Before



After



Zagrożenie rebase

- to są w zasadzie nowe commity, a nie przesunięcie starych - tworzymy zupełnie nową historię



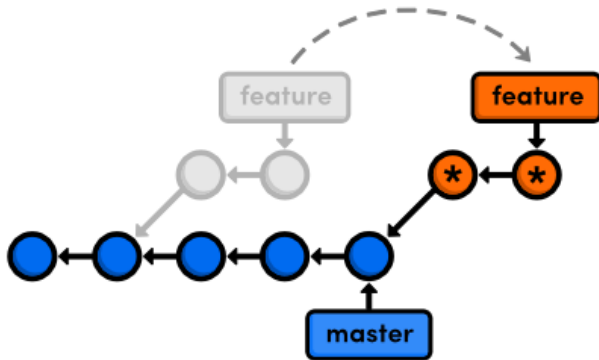
Zagrożenie rebase

- to są w zasadzie nowe commity, a nie przesunięcie starych - tworzymy zupełnie nową historię
- jeśli commity były współużytkowane, to po prostu znikają z historii dla innych użytkowników, co trudno jest rozwiązać lub cofnąć

Zagrożenie rebase

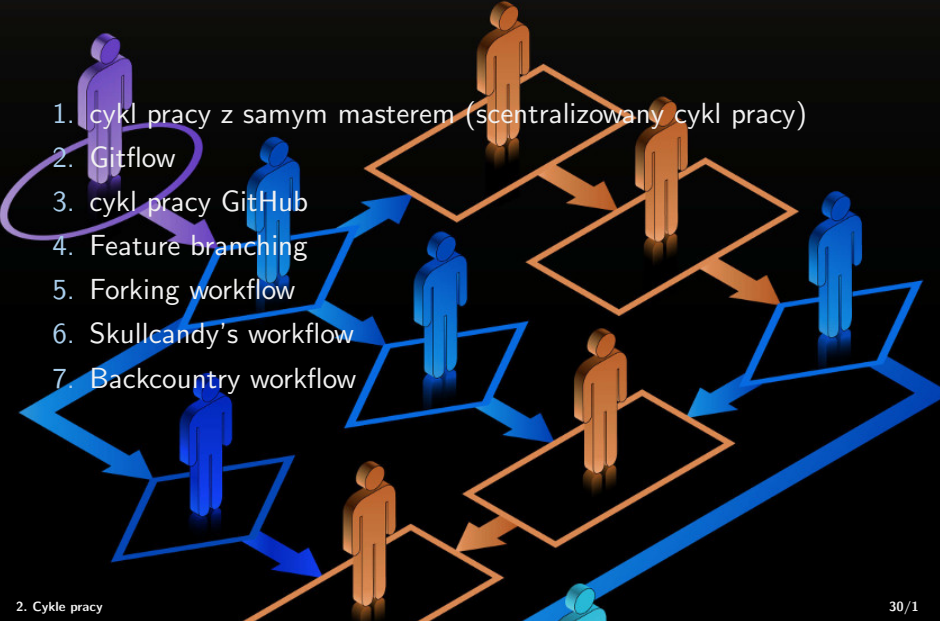
- to są w zasadzie nowe commity, a nie przesunięcie starych - tworzymy zupełnie nową historię
- jeśli commity były współużytkowane, to po prostu znikają z historii dla innych użytkowników, co trudno jest rozwiązać lub cofnąć
- nie należy rebase'ować gałęzi, jeśli jej commity były współdzielone

Faktyczny rebase



★ = Brand New Commits

Cykle pracy

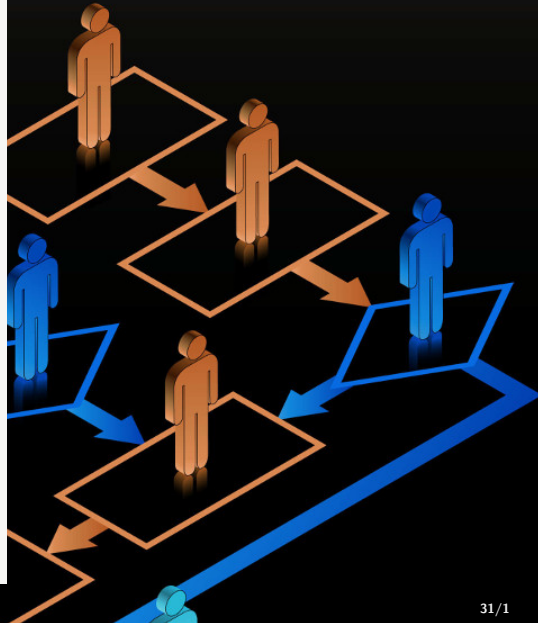
- 
- A diagram illustrating various work cycles. It features several stylized human figures in blue and orange, connected by a network of blue and orange arrows. The arrows form a complex web, with some pointing towards the center and others pointing outwards. A purple oval highlights the first three items in the list, and a purple arrow points from it to the first item.
1. cykl pracy z samym masterem (scentralizowany cykl pracy)
 2. Gitflow
 3. cykl pracy GitHub
 4. Feature branching
 5. Forking workflow
 6. Skullcandy's workflow
 7. Backcountry workflow

Scentralizowany

Master only

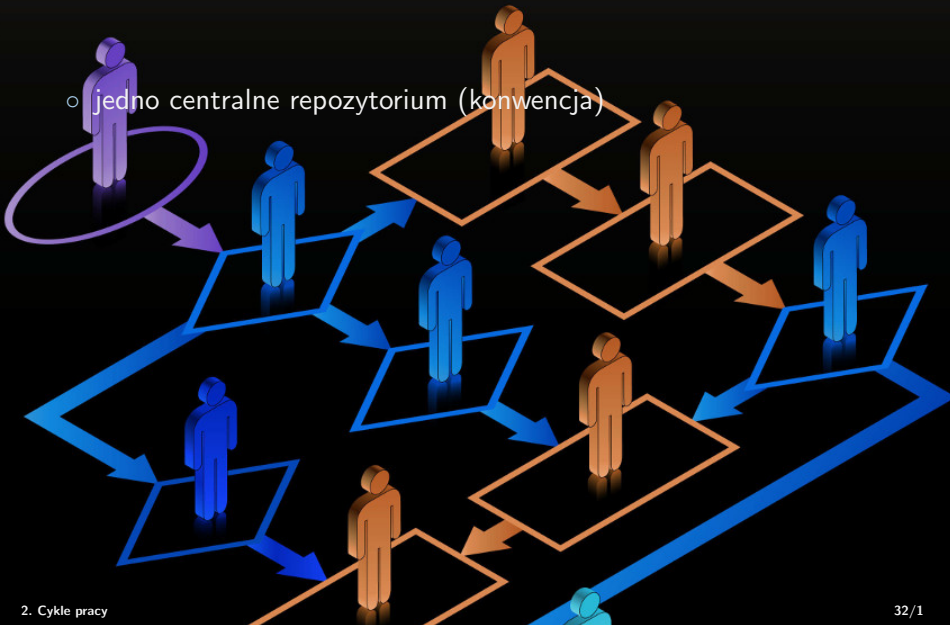


Master



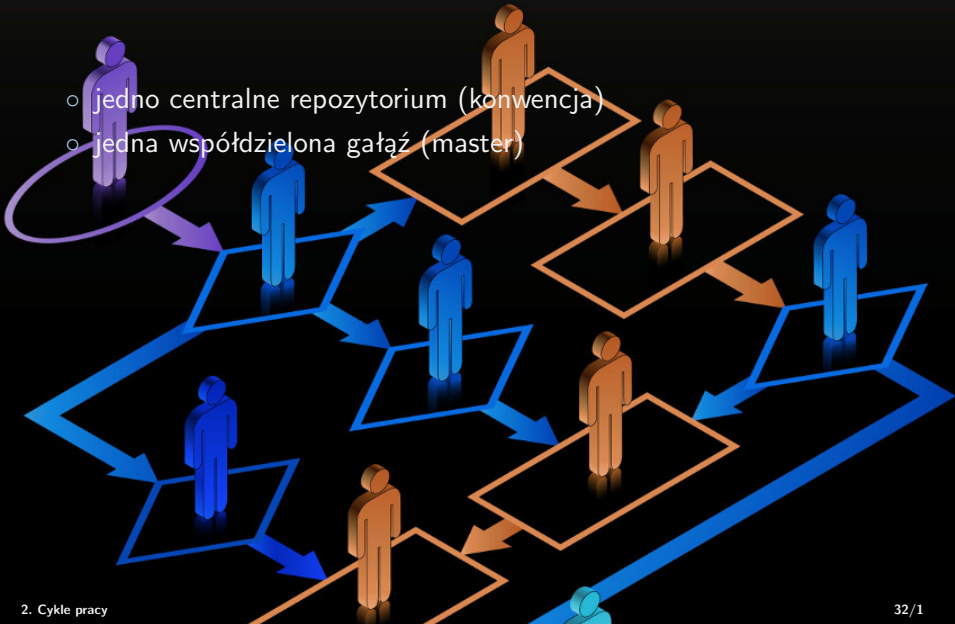
Scentralizowany

- o jedno centralne repozytorium (konwencja)



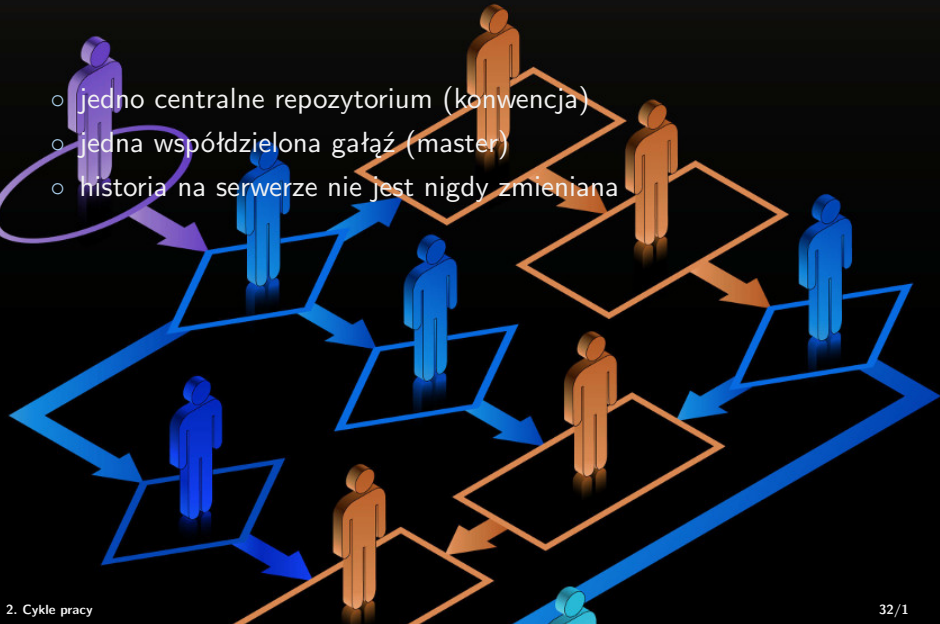
Scentralizowany

- o jedno centralne repozytorium (konwencja)
- o jedna współdzielona gałąź (master)



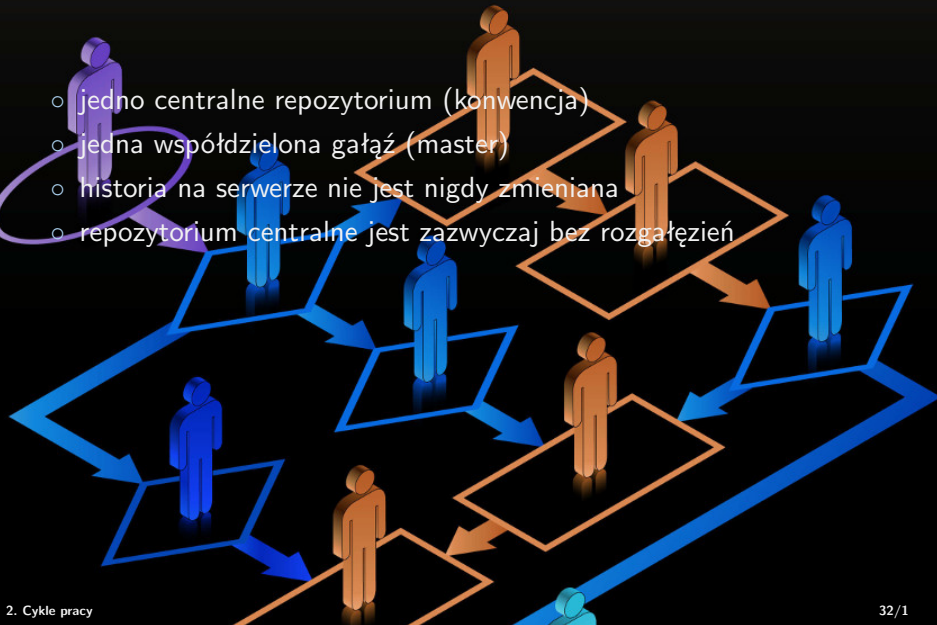
Scentralizowany

- o jedno centralne repozytorium (konwencja)
- o jedna współdzielona gałąź (master)
- o historia na serwerze nie jest nigdy zmieniana



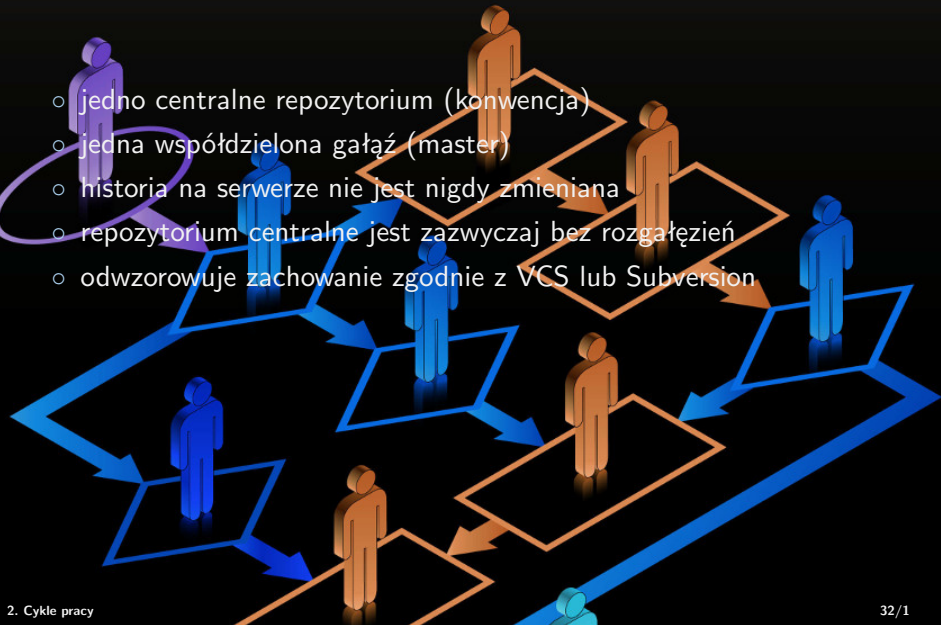
Scentralizowany

- o jedno centralne repozytorium (konwencja)
- o jedna współdzielona gałąź (master)
- o historia na serwerze nie jest nigdy zmieniana
- o repozytorium centralne jest zazwyczaj bez rozgałęzień



Scentralizowany

- jedno centralne repozytorium (konwencja)
- jedna współdzielona gałąź (master)
- historia na serwerze nie jest nigdy zmieniana
- repozytorium centralne jest zazwyczaj bez rozgałęzień
- odwzorowuje zachowanie zgodnie z VCS lub Subversion



Scentralizowany

- o jedno centralne repozytorium (konwencja)
- o jedna współdzielona gałąź (master)
- o historia na serwerze nie jest nigdy zmieniana
- o repozytorium centralne jest zazwyczaj bez rozgałęzień
- o odwzorowuje zachowanie zgodnie z VCS lub Subversion
- o lokalny master śledzi origin/master



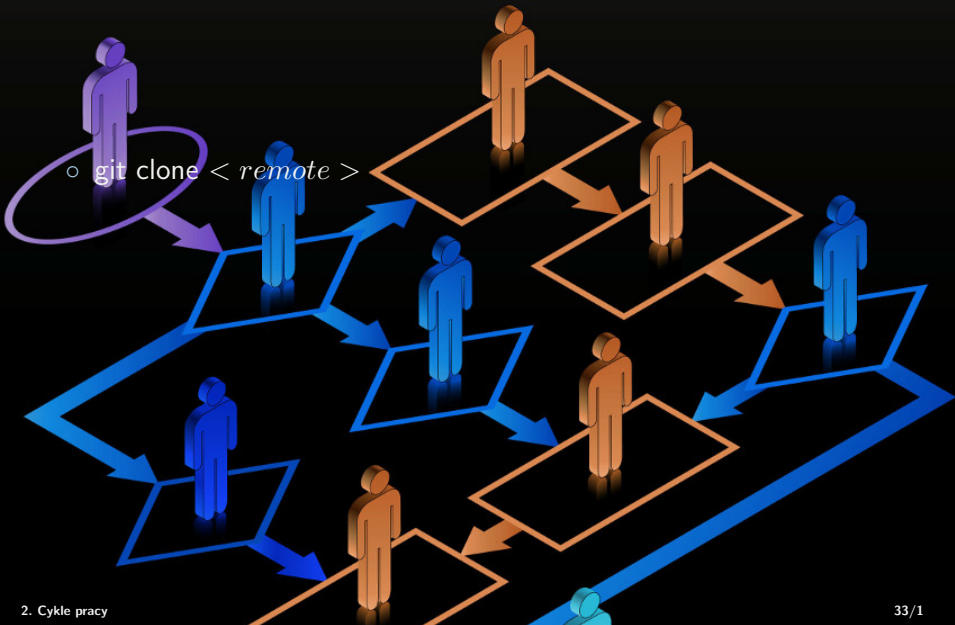
Scentralizowany

- jedno centralne repozytorium (konwencja)
- jedna współdzielona gałąź (master)
- historia na serwerze nie jest nigdy zmieniana
- repozytorium centralne jest zazwyczaj bez rozgałęzień
- odwzorowuje zachowanie zgodnie z VCS lub Subversion
- lokalny master śledzi origin/master
- preferowany jest lokalny rebase względem merge'a

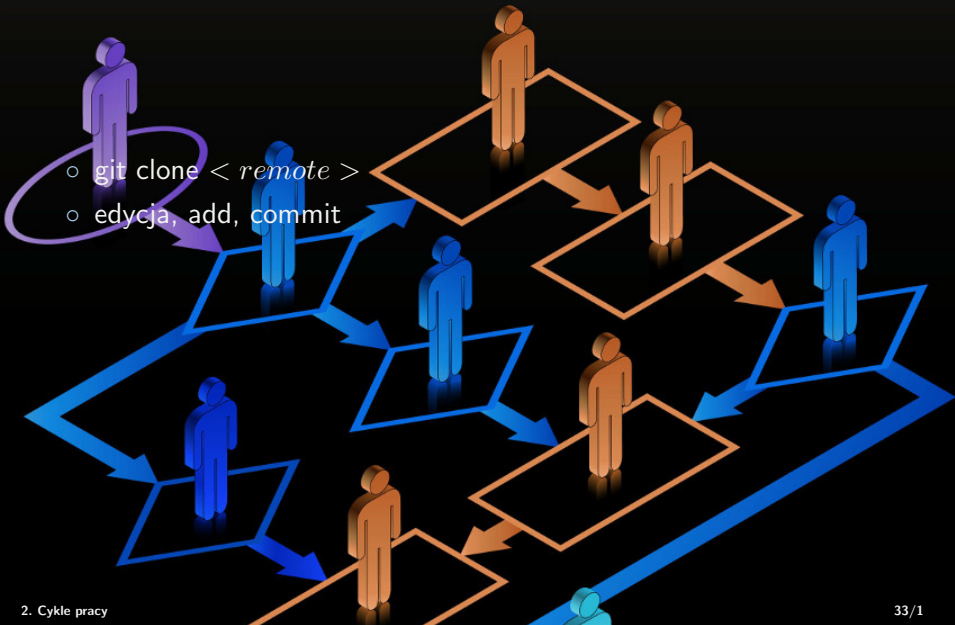
Scentralizowany

- jedno centralne repozytorium (konwencja)
- jedna współdzielona gałąź (master)
- historia na serwerze nie jest nigdy zmieniana
- repozytorium centralne jest zazwyczaj bez rozgałęzień
- odwzorowuje zachowanie zgodnie z VCS lub Subversion
- lokalny master śledzi origin/master
- preferowany jest lokalny rebase względem merge'a
- lokalne repozytorium zawsze ma gałęzie

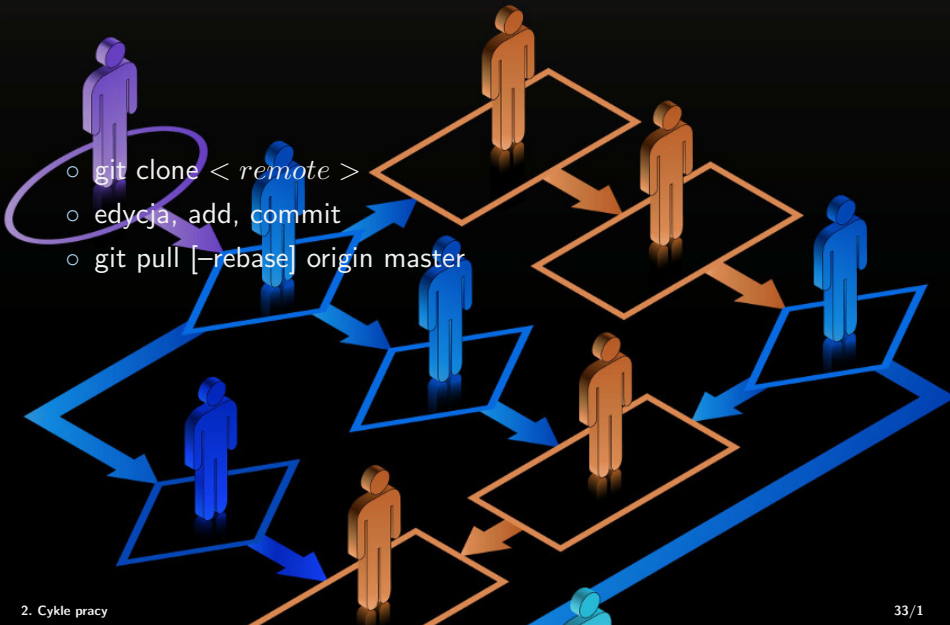
Typowy scenariusz dla cyklu Master



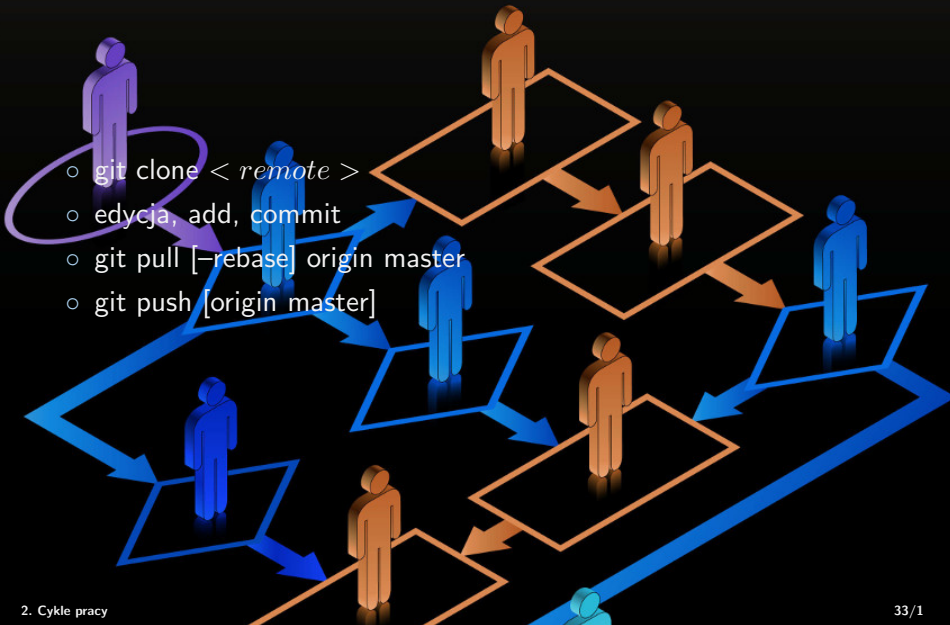
Typowy scenariusz dla cyklu Master



Typowy scenariusz dla cyklu Master

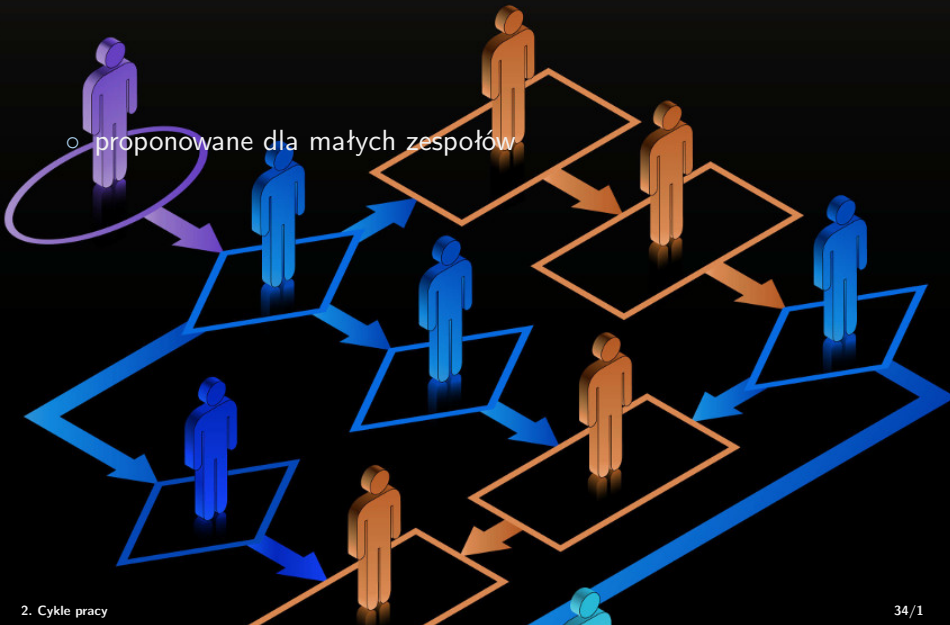
- 
- The diagram illustrates a distributed workflow for the Master cycle. It features several stylized human figures in blue and orange, each standing on a rectangular platform. These platforms are interconnected by a network of blue and orange arrows, representing the flow of work and code. A purple circle highlights the first figure, which is associated with the initial steps of the cycle: cloning, editing, committing, and pulling. The workflow shows a sequence of tasks being passed between team members, with some paths leading back to the origin, indicating a pull or merge operation.
- `git clone <remote>`
 - edycja, add, commit
 - `git pull [-rebase] origin master`

Typowy scenariusz dla cyklu Master

- 
- The diagram illustrates the Master cycle workflow. It features a central figure (purple) and several other figures (orange and blue) arranged in a circular pattern. Arrows indicate the flow of work: from the central figure to the orange figures, then to the blue figures, and finally back to the central figure. The arrows are color-coded: orange for the first step, blue for the second, and purple for the third. The central figure is also circled in purple.
- `git clone <remote>`
 - edycja, add, commit
 - `git pull [-rebase] origin master`
 - `git push [origin master]`

Master

- proponowane dla małych zespołów



Master

- proponowane dla małych zespołów
- kilka commitów wypychanych jednocześnie - istnieje ryzyko utraty, gdyż zmiany przed wypchnięciem są tylko lokalne. Dłuższa praca nad zadaniem również zwiększa to ryzyko.



Master

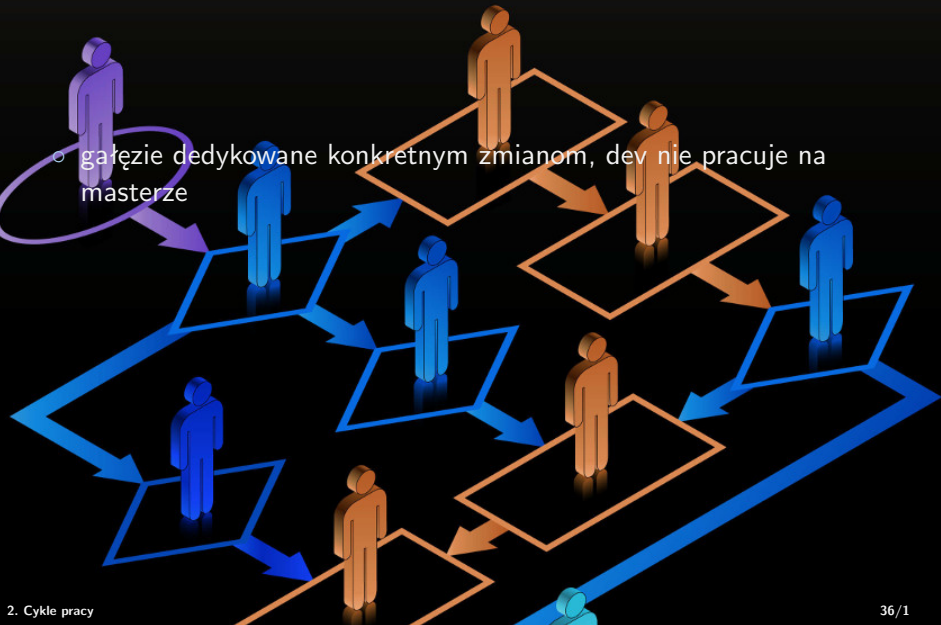
- proponowane dla małych zespołów
- kilka commitów wypychanych jednocześnie - istnieje ryzyko utraty, gdyż zmiany przed wypchnięciem są tylko lokalne. Dłuższa praca nad zadaniem również zwiększa to ryzyko.
- większe zespoły - często konflikty, trudność z zachowaniem liniowości mastera, utrudniona komunikacja (zmiany tylko lokalne)

Feature branch



Feature branch

- o gałęzie dedykowane konkretnym zmianom, dev nie pracuje na masterze



Feature branch

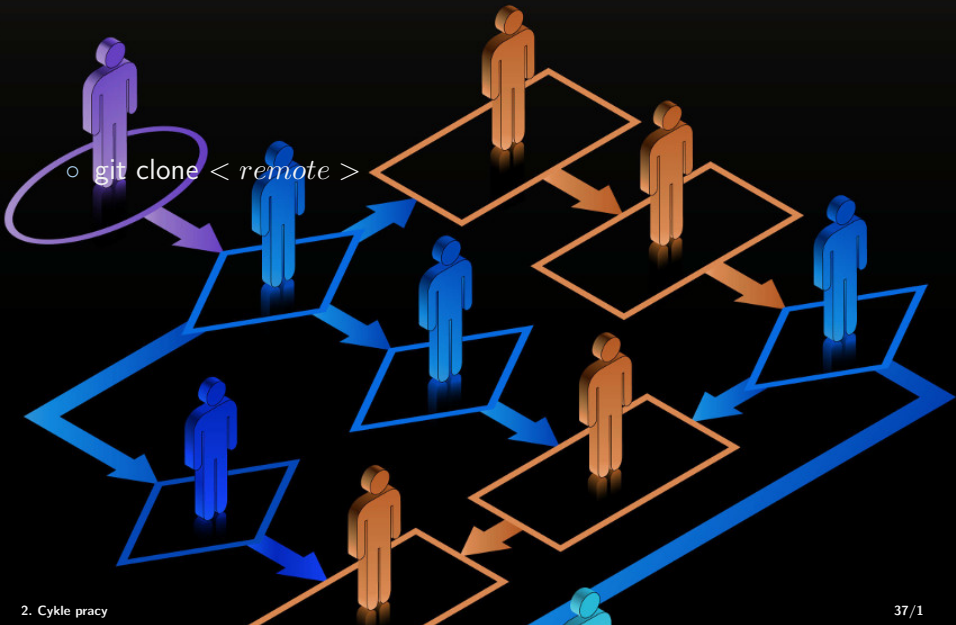
- o gałęzie dedykowane konkretnym zmianom, dev nie pracuje na masterze
- o często rozszerza scentralizowany cykl pracy (rebase)



Feature branch

- gałęzie dedykowane konkretnym zmianom, dev nie pracuje na masterze
- często rozszerza scentralizowany cykl pracy (rebase)
- gałęzie dedykowane są wypychane na serwer (współpraca przy jednej gałęzi)

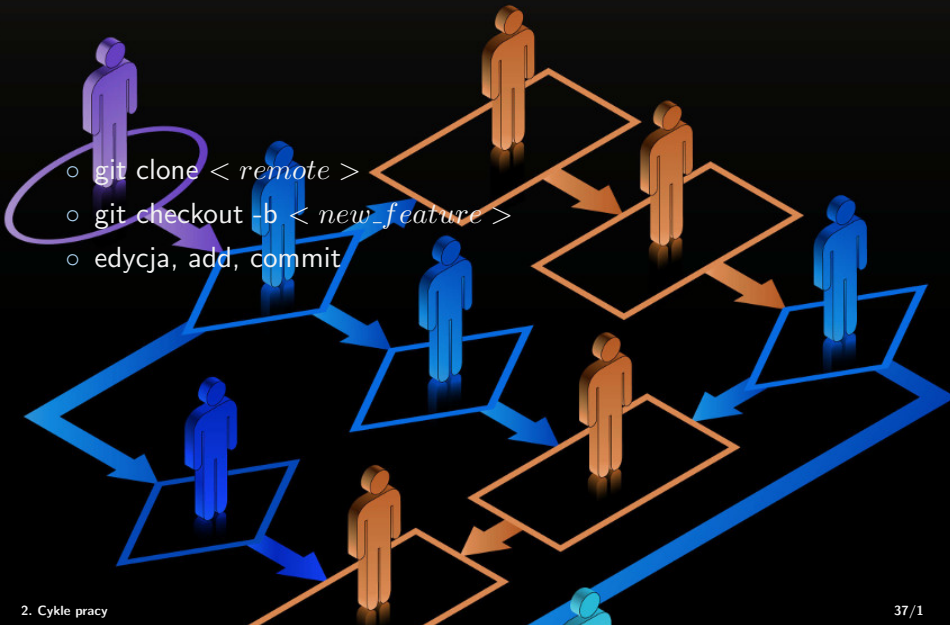
Typowy scenariusz dla cyklu Feature branch



Typowy scenariusz dla cyklu Feature branch

- `git clone <remote>`
 - `git checkout -b <new_feature>`
- 
- A diagram illustrating a typical Feature Branch workflow. It features several stylized human figures in blue and orange, each standing on a rectangular platform. The platforms are interconnected by a network of blue and orange arrows, representing the flow of code and work. A purple oval highlights the first two steps of the workflow: 'git clone <remote>' and 'git checkout -b <new_feature>'. The workflow starts with a purple figure, moves to a blue figure, then to an orange figure, and continues through a series of blue and orange figures, eventually leading to a light blue figure at the bottom. The arrows indicate the sequence of operations, such as cloning, branching, and merging.

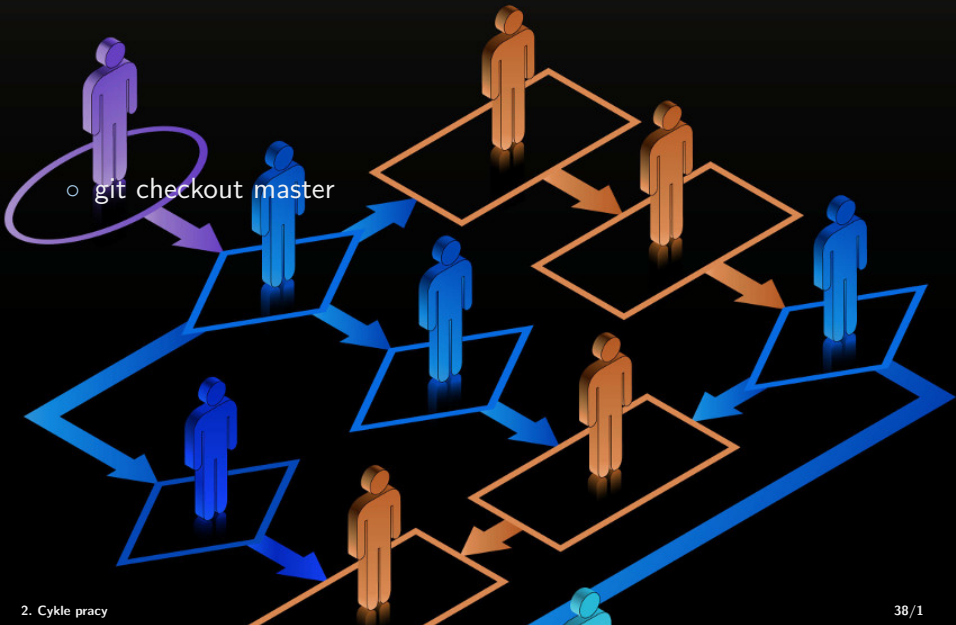
Typowy scenariusz dla cyklu Feature branch

- git clone <remote>
 - git checkout -b <new_feature>
 - edycja, add, commit
- 
- A diagram illustrating a typical Feature Branch workflow. It features several stylized human figures in blue and orange, each standing on a rectangular platform. The platforms are interconnected by a network of blue and orange arrows, representing the flow of code and work. A purple circle highlights the first three steps of the workflow: 'git clone <remote>', 'git checkout -b <new_feature>', and 'edycja, add, commit'. The workflow starts with a blue figure on a platform, moves to an orange figure, then to another blue figure, and continues through a series of orange and blue figures, eventually leading to a final blue figure at the bottom right. The arrows indicate the sequence of operations and the branching strategy used in the development process.

Typowy scenariusz dla cyklu Feature branch

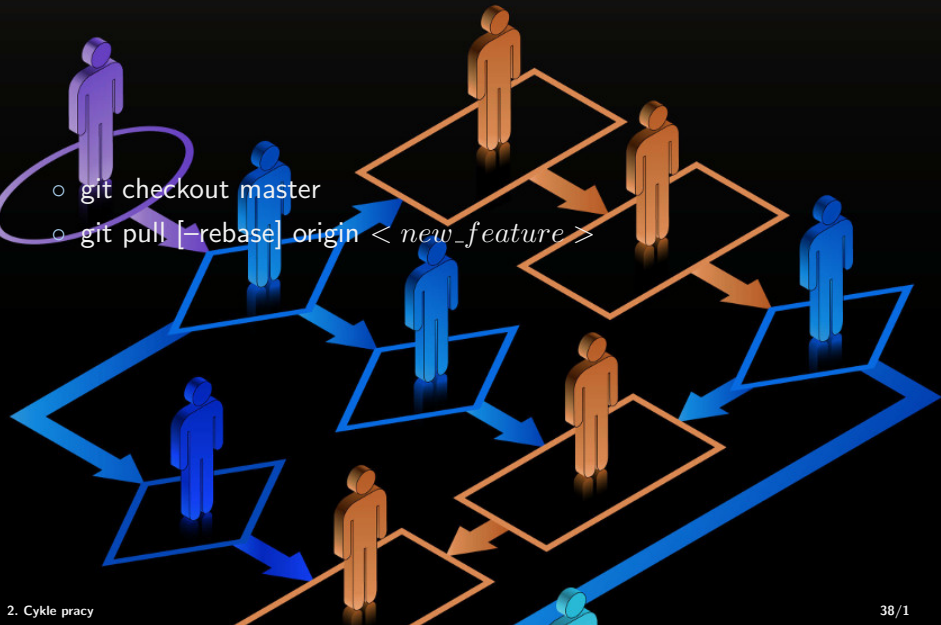
- 
- The diagram illustrates a typical Feature Branch workflow. It features several stylized human figures in blue and orange, each standing on a rectangular platform. The platforms are interconnected by a network of blue and orange arrows. A large blue arrow forms a loop, starting from a blue figure on the left, passing through several orange figures, and returning to the start. Smaller orange arrows connect the platforms in a sequence, showing the flow of development. A purple circle highlights the first blue figure, and a purple arrow points from it to the first item in the list.
- `git clone <remote>`
 - `git checkout -b <new_feature>`
 - edycja, add, commit
 - `git push [-u origin new_feature]`

Feature branch (merge do mastera)

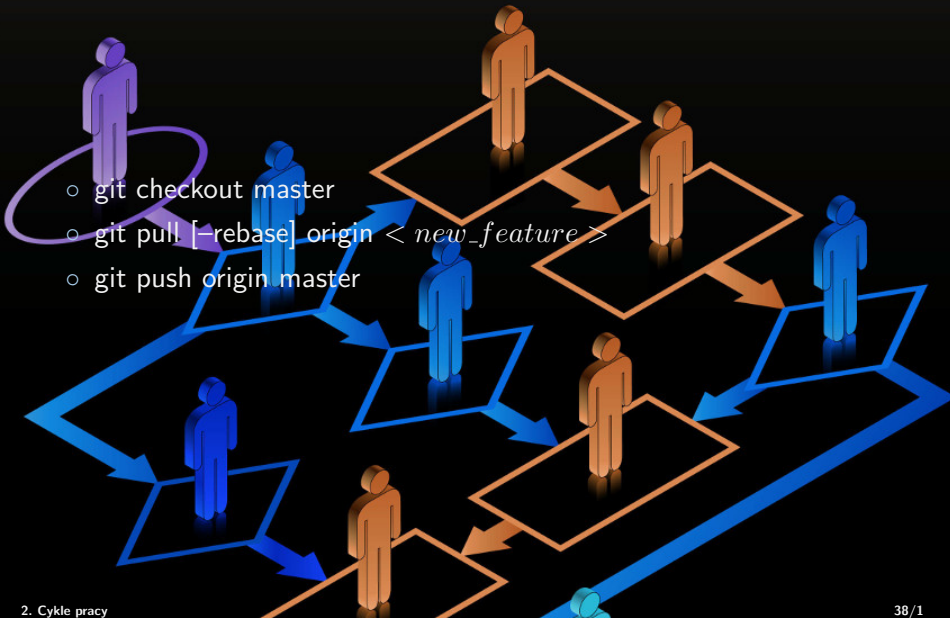


Feature branch (merge do mastera)

- git checkout master
- git pull [-rebase] origin < *new_feature* >



Feature branch (merge do mastera)

- git checkout master
 - git pull [-rebase] origin < *new_feature* >
 - git push origin master
- 

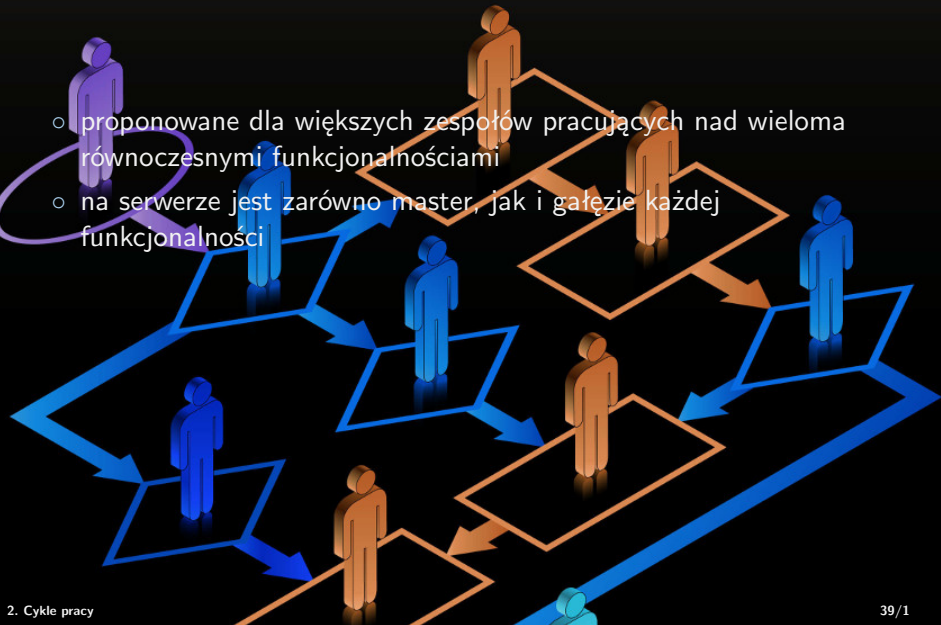
Feature branch

- proponowane dla większych zespołów pracujących nad wieloma równoczesnymi funkcjonalnościami



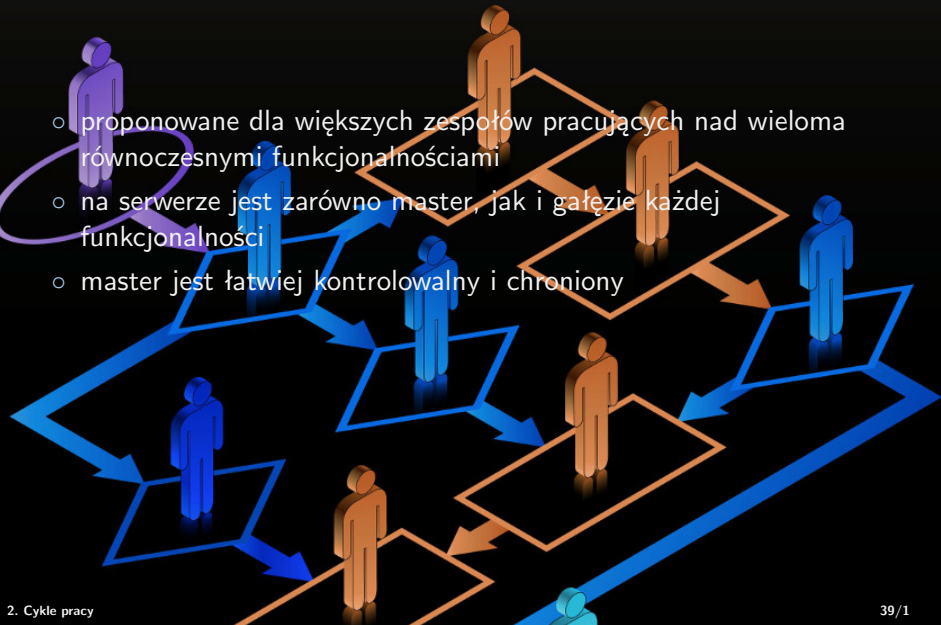
Feature branch

- proponowane dla większych zespołów pracujących nad wieloma równoczesnymi funkcjonalnościami
- na serwerze jest zarówno master, jak i gałęzie każdej funkcjonalności



Feature branch

- proponowane dla większych zespołów pracujących nad wieloma równoczesnymi funkcjonalnościami
- na serwerze jest zarówno master, jak i gałęzie każdej funkcjonalności
- master jest łatwiej kontrolowalny i chroniony



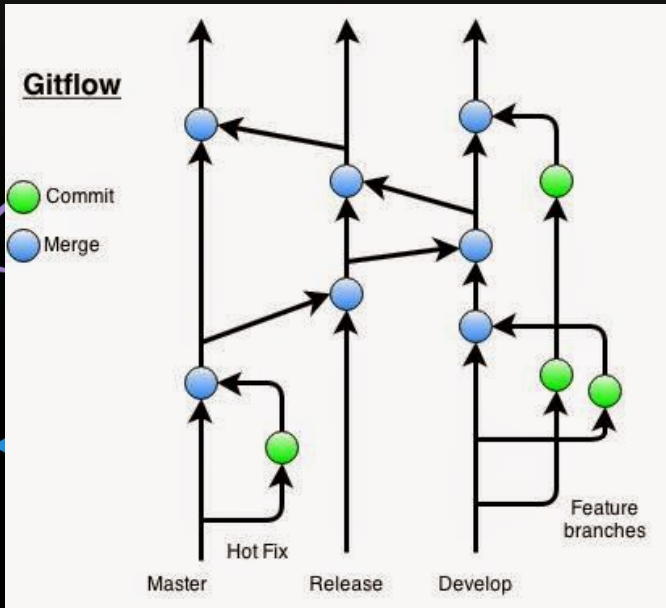
Feature branch

- proponowane dla większych zespołów pracujących nad wieloma równoczesnymi funkcjonalnościami
- na serwerze jest zarówno master, jak i gałęzie każdej funkcjonalności
- master jest łatwiej kontrolowalny i chroniony
- elastyczny cykl pracy dla odpowiedzialnych programistów

Feature branch

- proponowane dla większych zespołów pracujących nad wieloma równoczesnymi funkcjonalnościami
- na serwerze jest zarówno master, jak i gałęzie każdej funkcjonalności
- master jest łatwiej kontrolowalny i chroniony
- elastyczny cykl pracy dla odpowiedzialnych programistów
- wspiera pull request, gałęzie eksperymentalne, wydajną współpracę

Gitflow

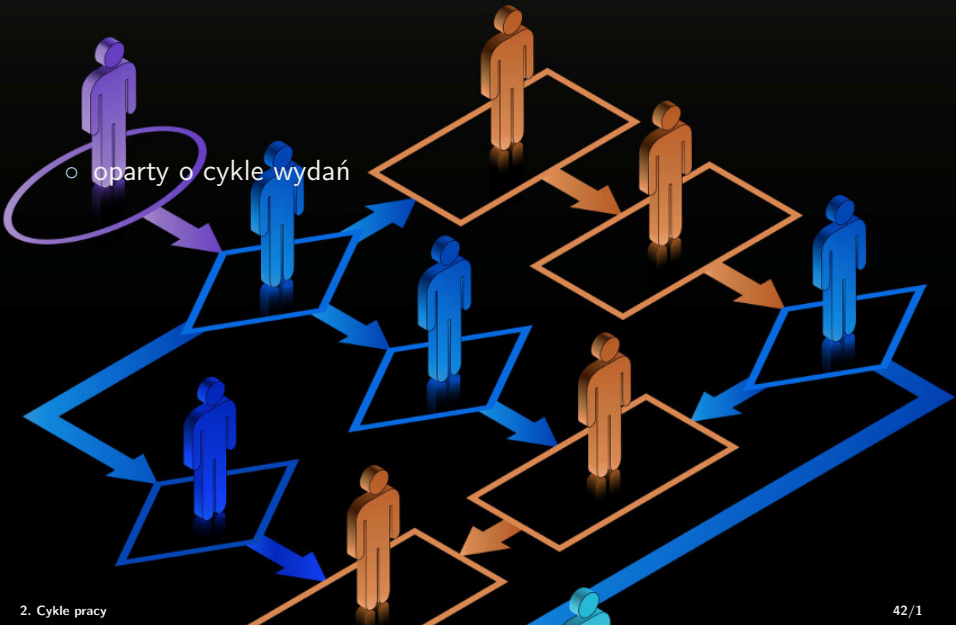


The Gitflow Workflow



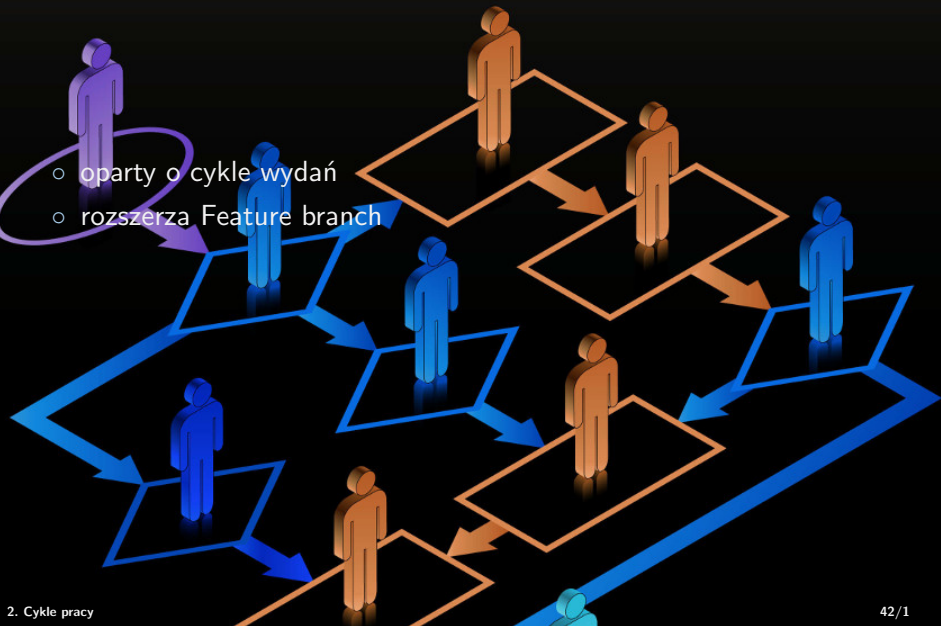
Image source: <https://www.atlassian.com/git/tutorials/workflow-gitflow>

Gitflow



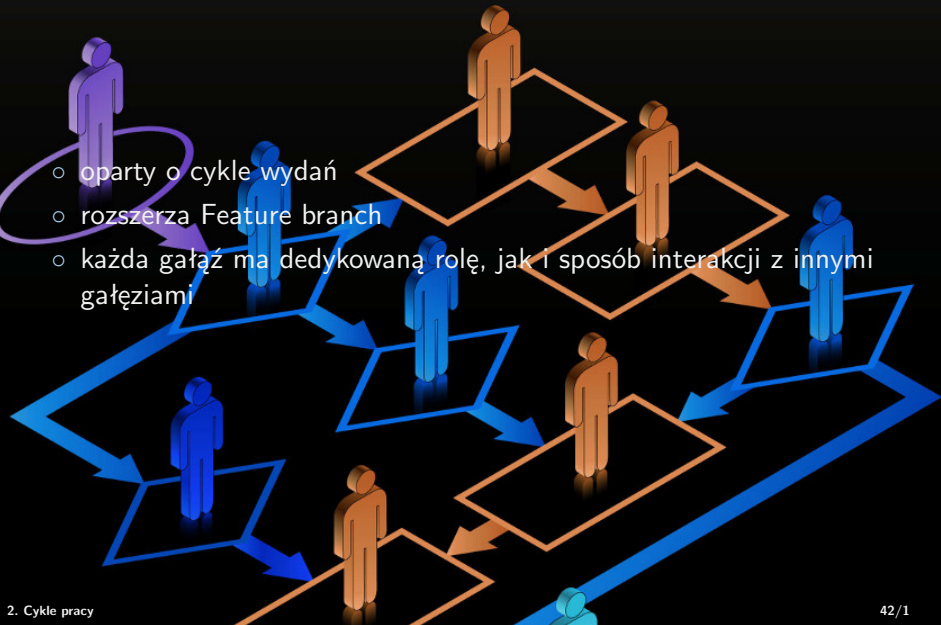
Gitflow

- o oparty o cykle wydań
- o rozszerza Feature branch



Gitflow

- o oparty o cykle wydań
- o rozszerza Feature branch
- o każda gałąź ma dedykowaną rolę, jak i sposób interakcji z innymi gałęziami



Gałęzie Gitflow

- master - oficjalne wydania, tagowanie, czasami w zaniku



Gałęzie Gitflow

- master - oficjalne wydania, tagowanie, czasami w zaniku
- hotfix - łatki, bugfixy - merge do mastera



Gałęzie Gitflow

- master - oficjalne wydania, tagowanie, czasami w zaniku
- hotfix - łatki, bugfixy - merge do mastera
- release - kandydat do wydania - merge do mastera



Gałęzie Gitflow

- master - oficjalne wydania, tagowanie, czasami w zaniku
- hotfix - łatki, bugfixy - merge do mastera
- release - kandydat do wydania - merge do mastera
- develop - gałąź integracyjna dla rozwijanych funkcjonalności - merge do release



Gałęzie Gitflow

- master - oficjalne wydania, tagowanie, czasami w zaniku
- hotfix - łatki, bugfixy - merge do mastera
- release - kandydat do wydania - merge do mastera
- develop - gałąź integracyjna dla rozwijanych funkcjonalności - merge do release
- feature - dedykowane gałęzie dla funkcjonalności - merge do develop

Gałęzie Gitflow - święty master

- tylko wybrani mogą do niego mergować



Gałęzie Gitflow - święty master

- tylko wybrani mogą do niego mergować
- tylko wybrane stabilne releasey mogą być mergowane



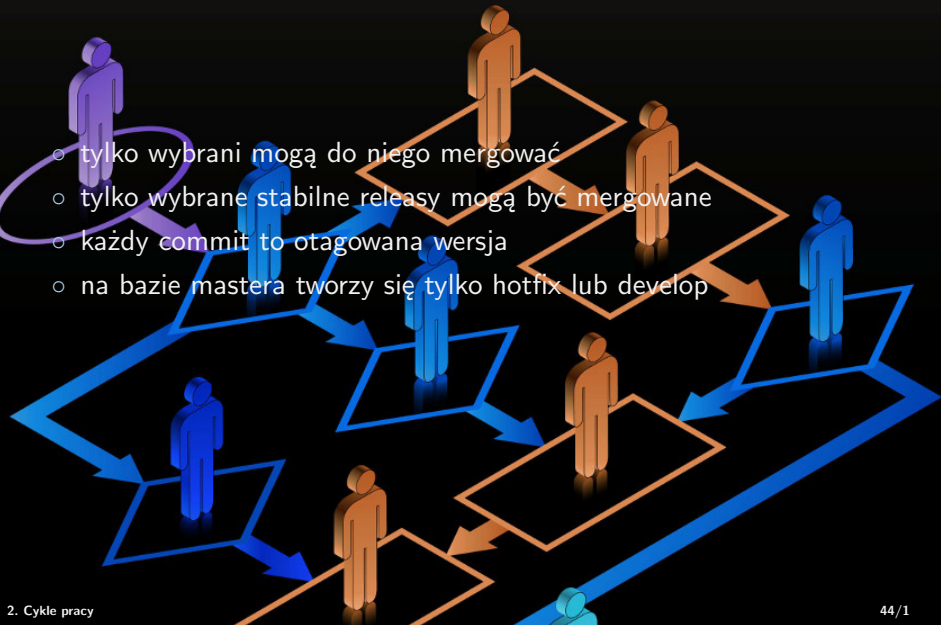
Gałęzie Gitflow - święty master

- tylko wybrani mogą do niego mergować
- tylko wybrane stabilne releasy mogą być mergowane
- każdy commit to otagowana wersja



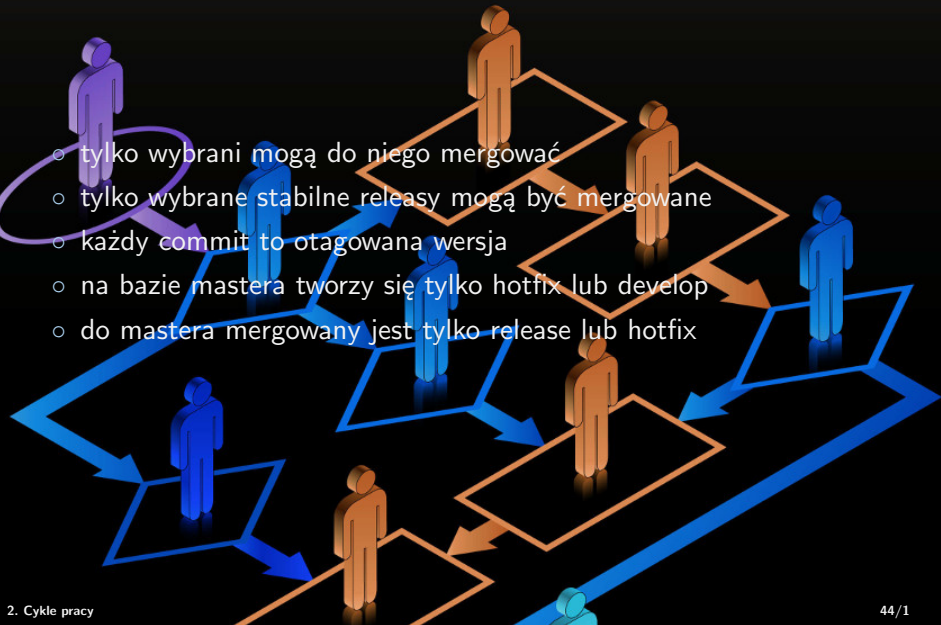
Gałęzie Gitflow - święty master

- tylko wybrani mogą do niego mergować
- tylko wybrane stabilne releasy mogą być mergowane
- każdy commit to otagowana wersja
- na bazie mastera tworzy się tylko hotfix lub develop



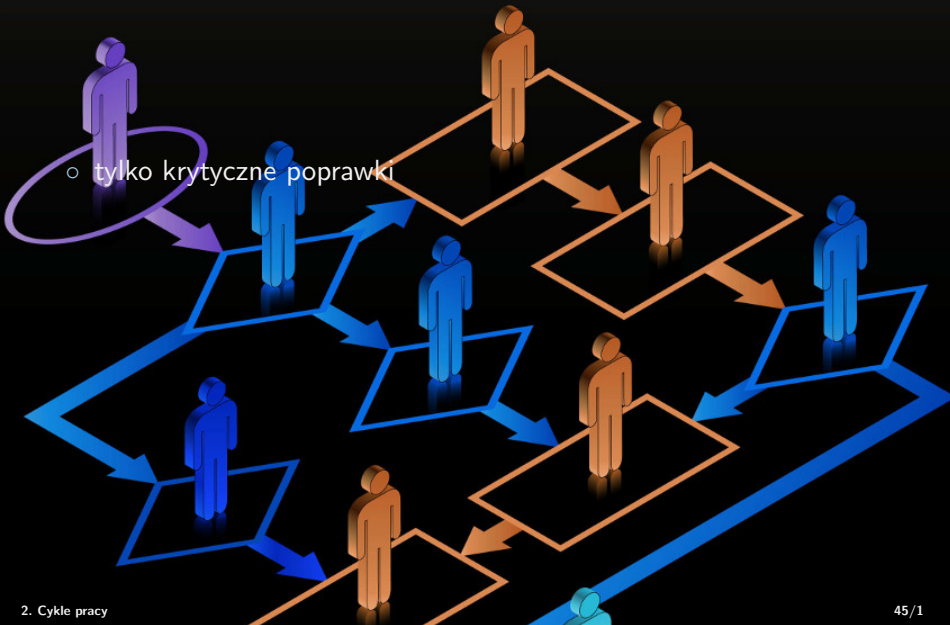
Gałęzie Gitflow - święty master

- tylko wybrani mogą do niego mergować
- tylko wybrane stabilne releasy mogą być mergowane
- każdy commit to otagowana wersja
- na bazie mastera tworzy się tylko hotfix lub develop
- do mastera mergowany jest tylko release lub hotfix



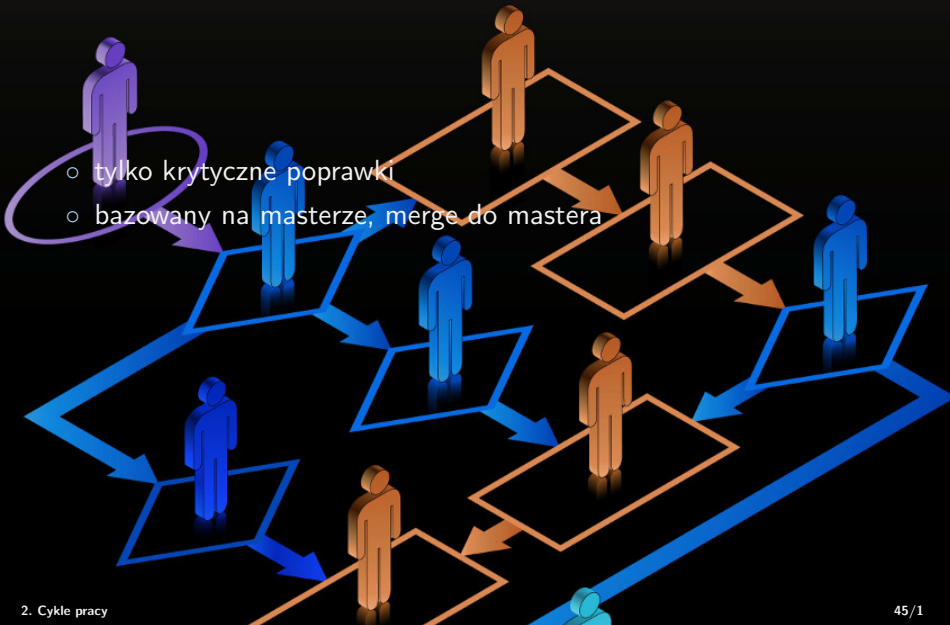
Gałęzie Gitflow - hotfix

- o tylko krytyczne poprawki



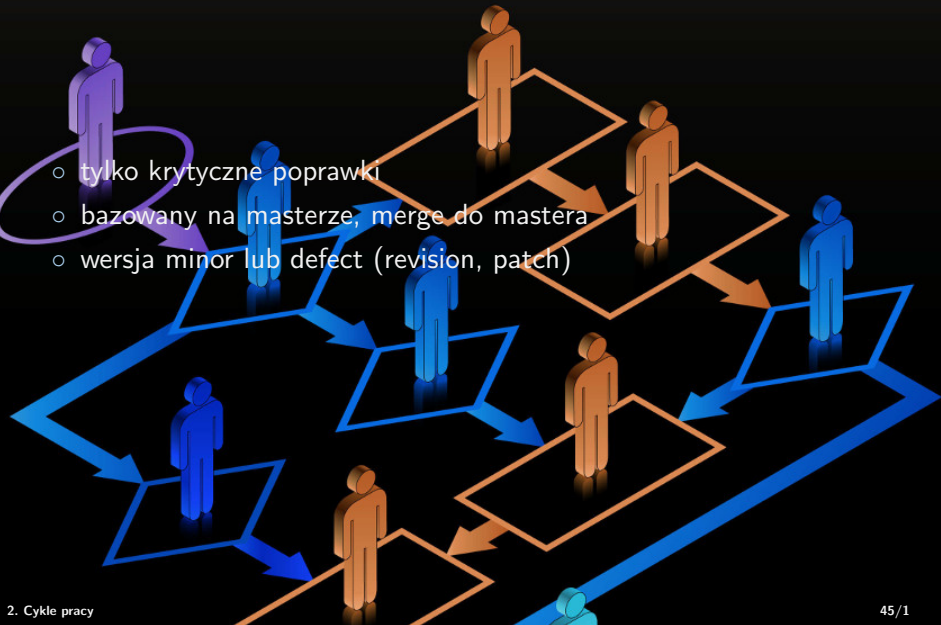
Gałęzie Gitflow - hotfix

- tylko krytyczne poprawki
- bazowany na masterze, merge do mastera



Gałęzie Gitflow - hotfix

- o tylko krytyczne poprawki
- o bazowany na masterze, merge do mastera
- o wersja minor lub defect (revision, patch)

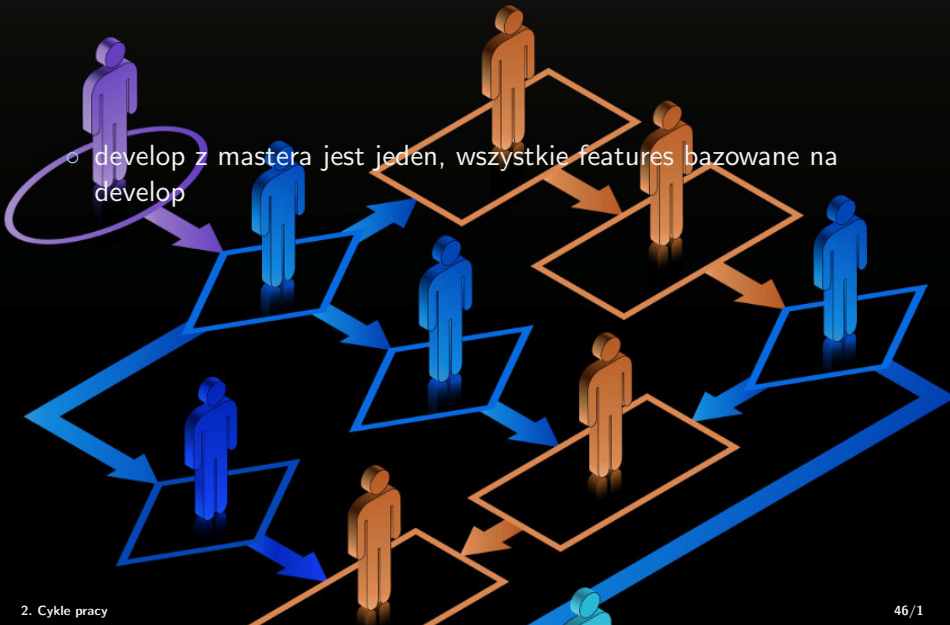


Gałęzie Gitflow - hotfix

- o tylko krytyczne poprawki
- o bazowany na masterze, merge do mastera
- o wersja minor lub defect (revision, patch)
- o występuje ryzyko używania jak gałęzi do zwykłego developmentu

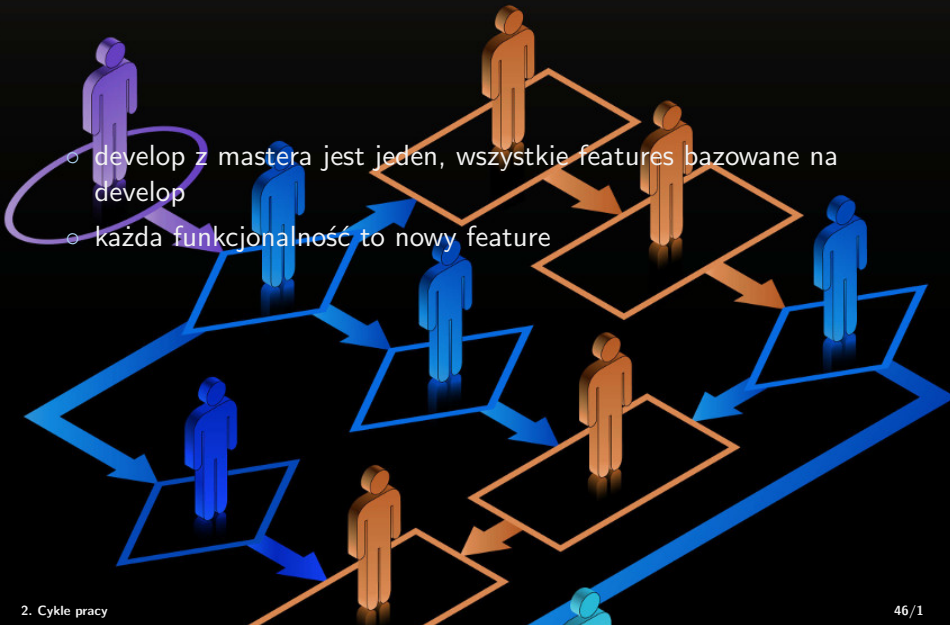
Gałęzie Gitflow - features

- develop z mastera jest jeden, wszystkie features bazowane na develop



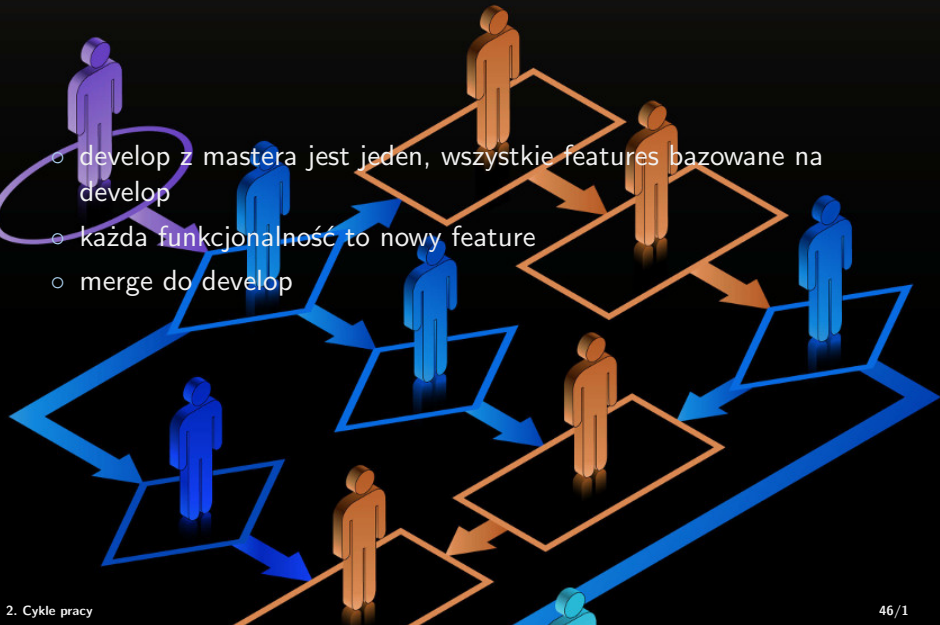
Gałęzie Gitflow - features

- develop z mastera jest jeden, wszystkie features bazowane na develop
- każda funkcjonalność to nowy feature



Gałęzie Gitflow - features

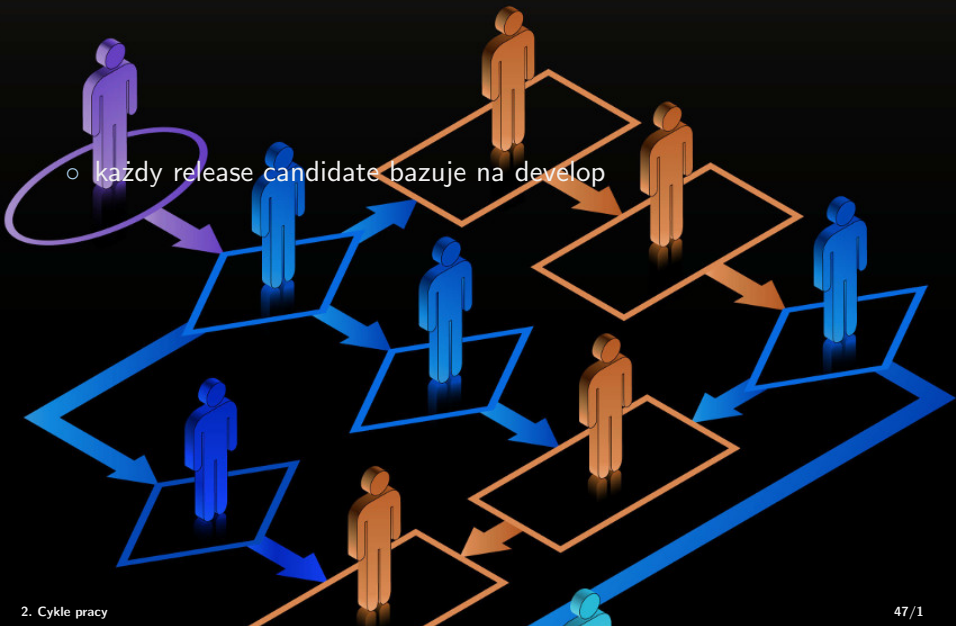
- develop z mastera jest jeden, wszystkie features bazowane na develop
- każda funkcjonalność to nowy feature
- merge do develop



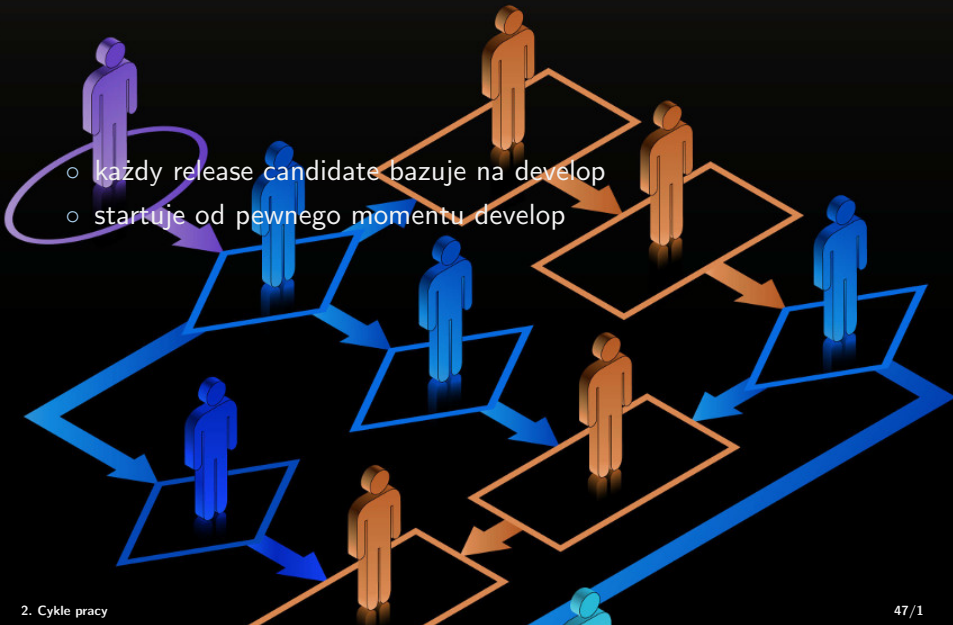
Gałęzie Gitflow - features

- develop z mastera jest jeden, wszystkie features bazowane na develop
- każda funkcjonalność to nowy feature
- merge do develop
- nazwa związana z funkcjonalnością

Gałęzie Gitflow - releases



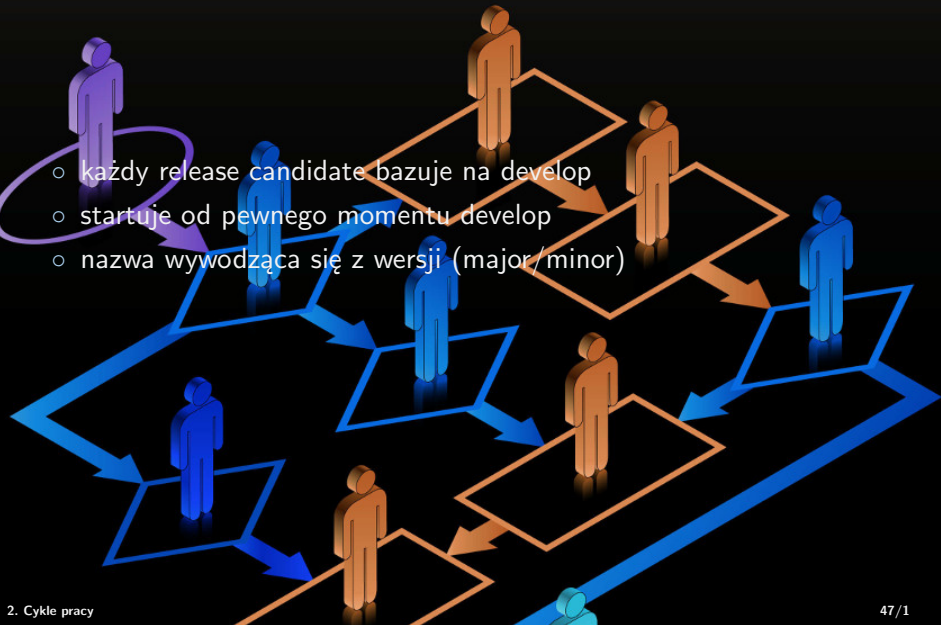
Gałęzie Gitflow - releases



- każdy release candidate bazuje na develop
- startuje od pewnego momentu develop

Gałęzie Gitflow - releases

- każdy release candidate bazuje na develop
- startuje od pewnego momentu develop
- nazwa wywodząca się z wersji (major/minor)



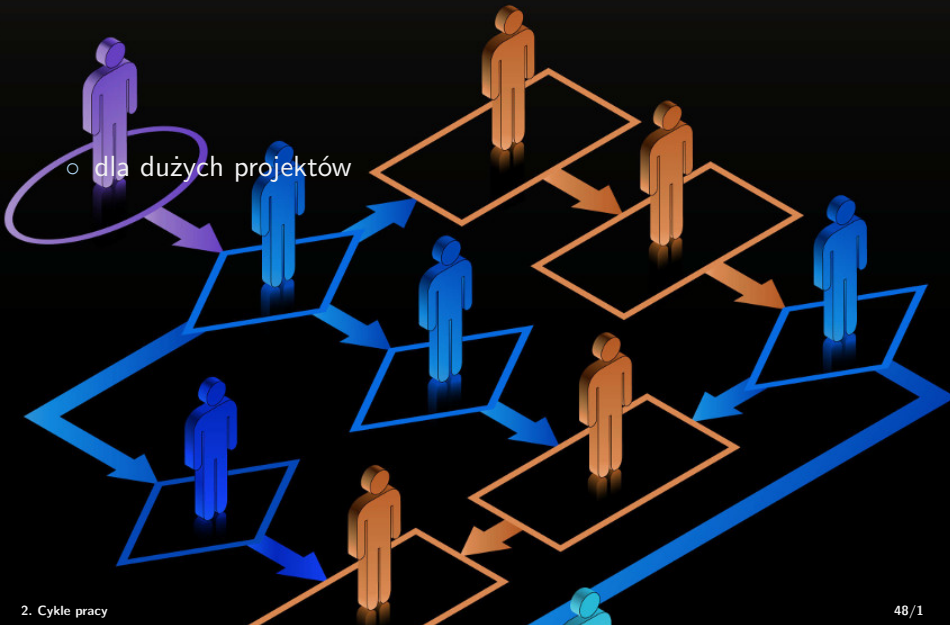
Gałęzie Gitflow - releases

- każdy release candidate bazuje na develop
- startuje od pewnego momentu develop
- nazwa wywodząca się z wersji (major/minor)
- merge do mastera i develop, gdy się kończy



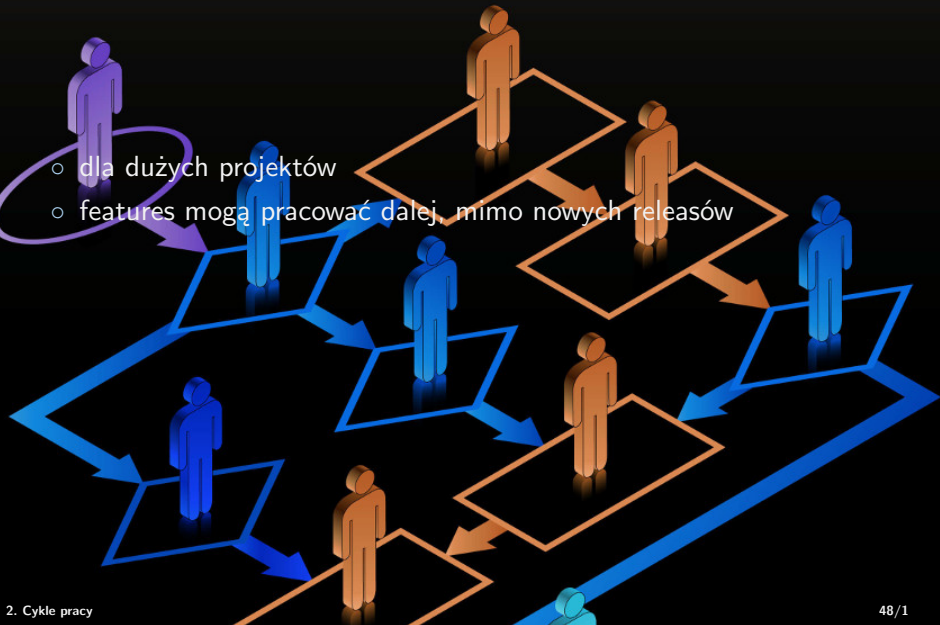
Gitflow

- dla dużych projektów



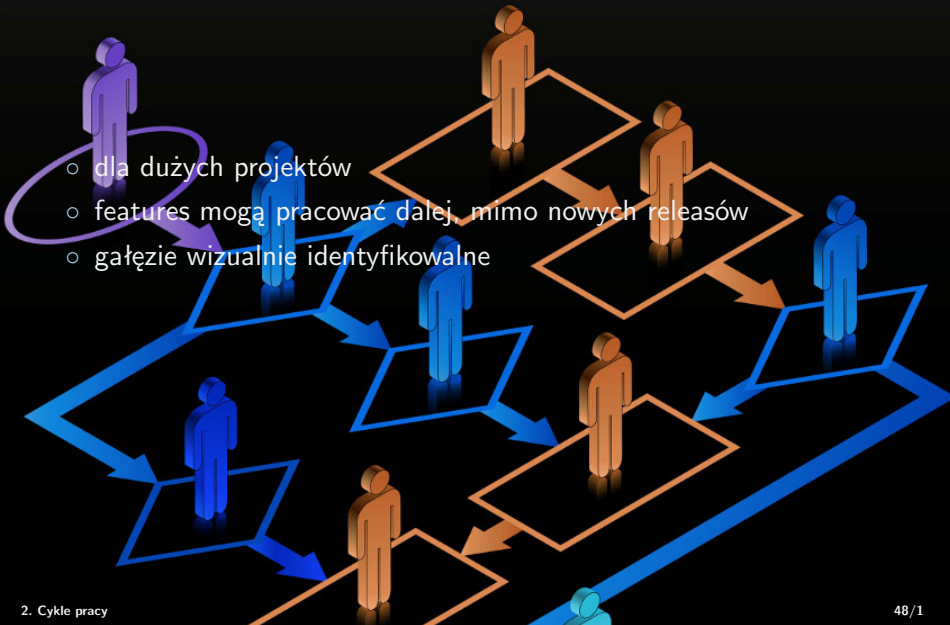
Gitflow

- dla dużych projektów
- features mogą pracować dalej, mimo nowych releasów



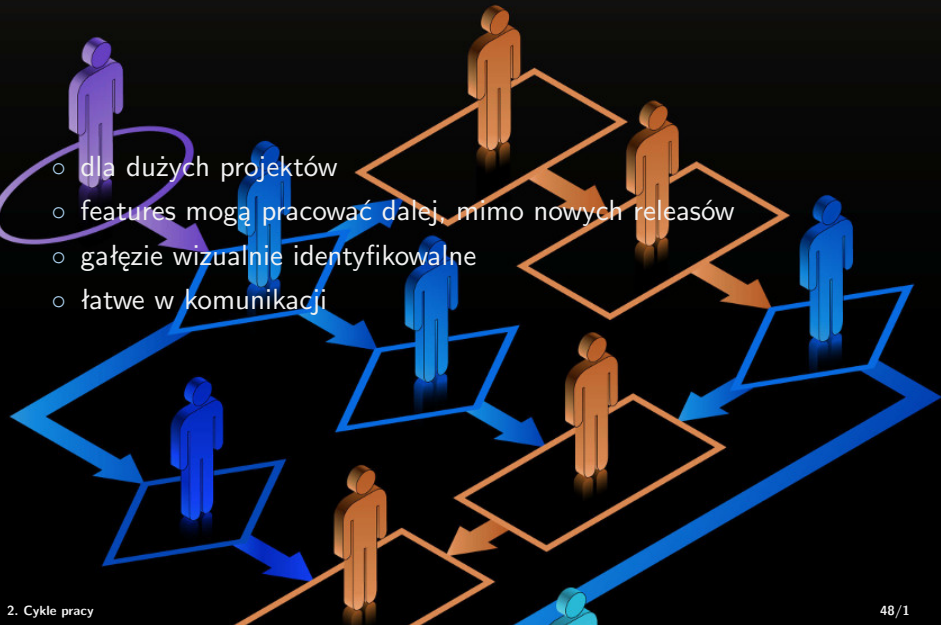
Gitflow

- dla dużych projektów
- features mogą pracować dalej, mimo nowych releasów
- gałęzie wizualnie identyfikowalne

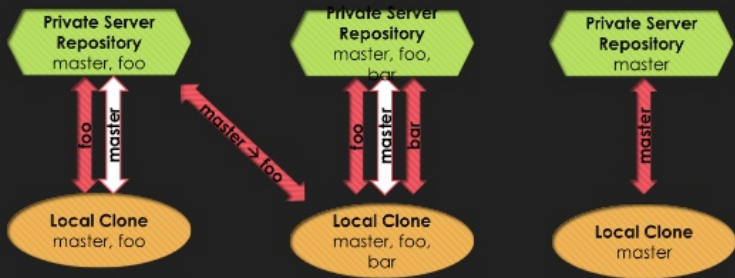


Gitflow

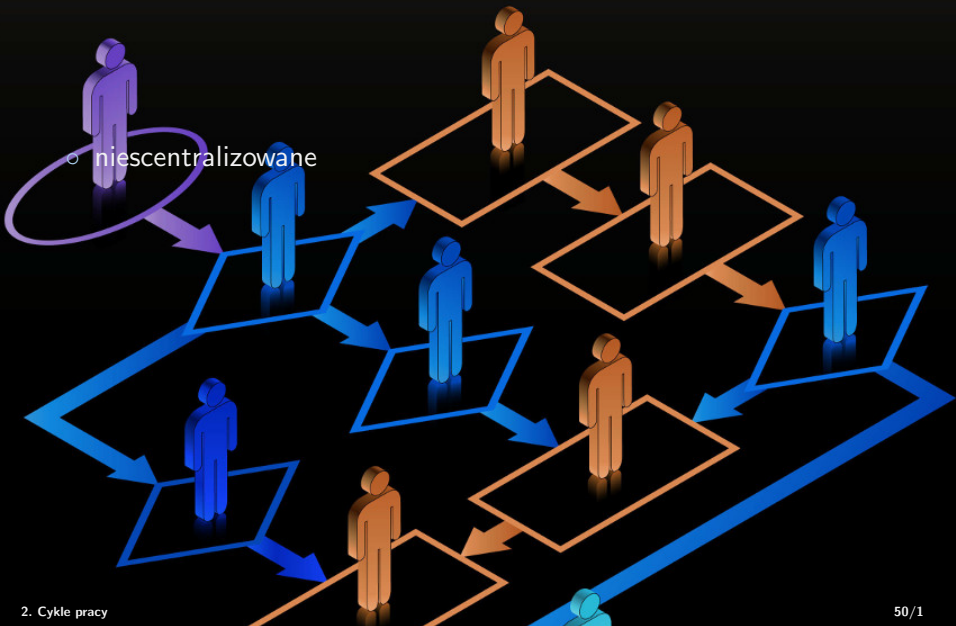
- dla dużych projektów
- features mogą pracować dalej, mimo nowych releasów
- gałęzie wizualnie identyfikowalne
- łatwe w komunikacji



The Forking Workflow

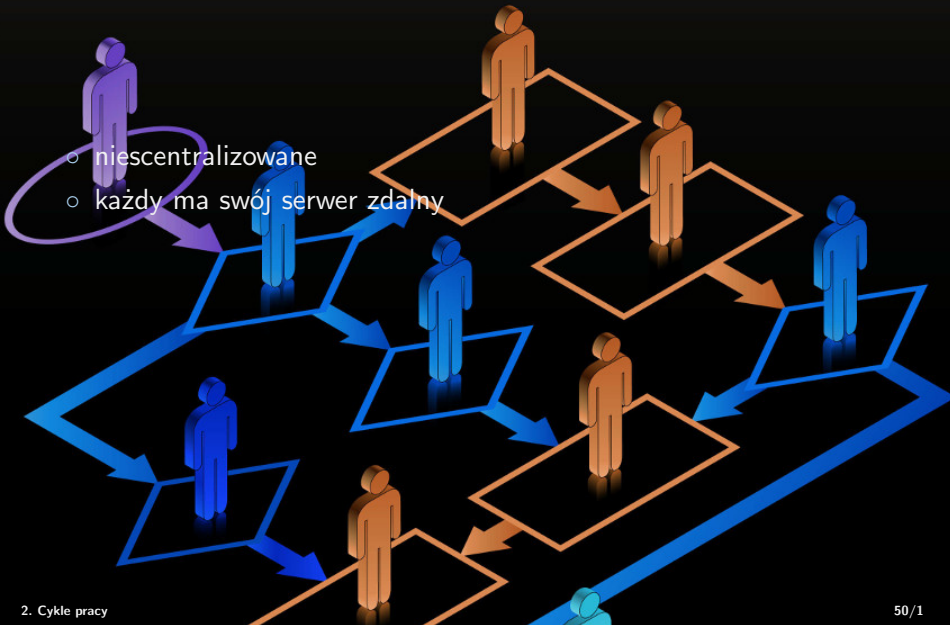


Forking workflow



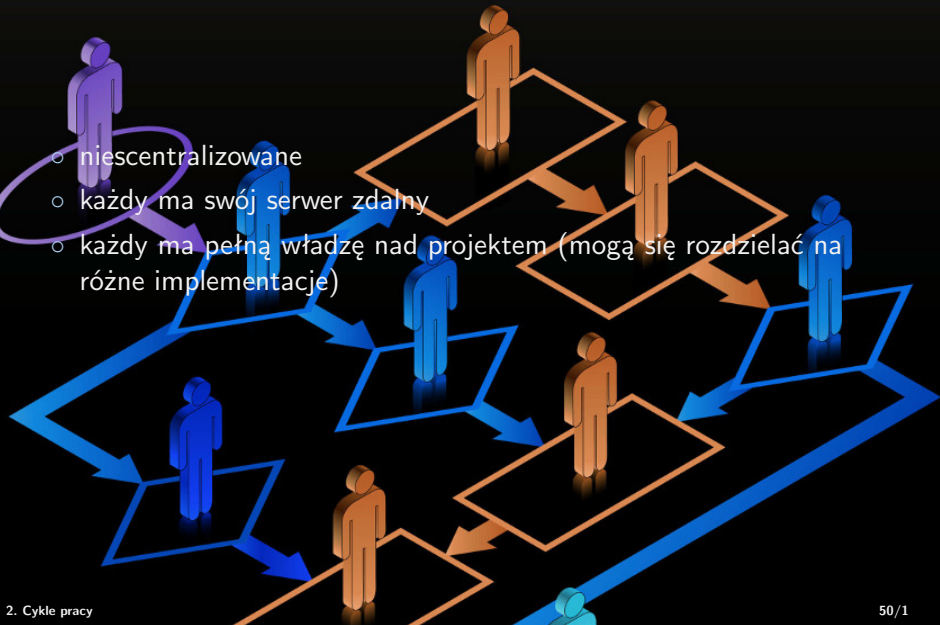
Forking workflow

- niescentralizowane
- każdy ma swój serwer zdalny



Forking workflow

- niescentralizowane
- każdy ma swój serwer zdalny
- każdy ma pełną władzę nad projektem (mogą się rozdzielać na różne implementacje)



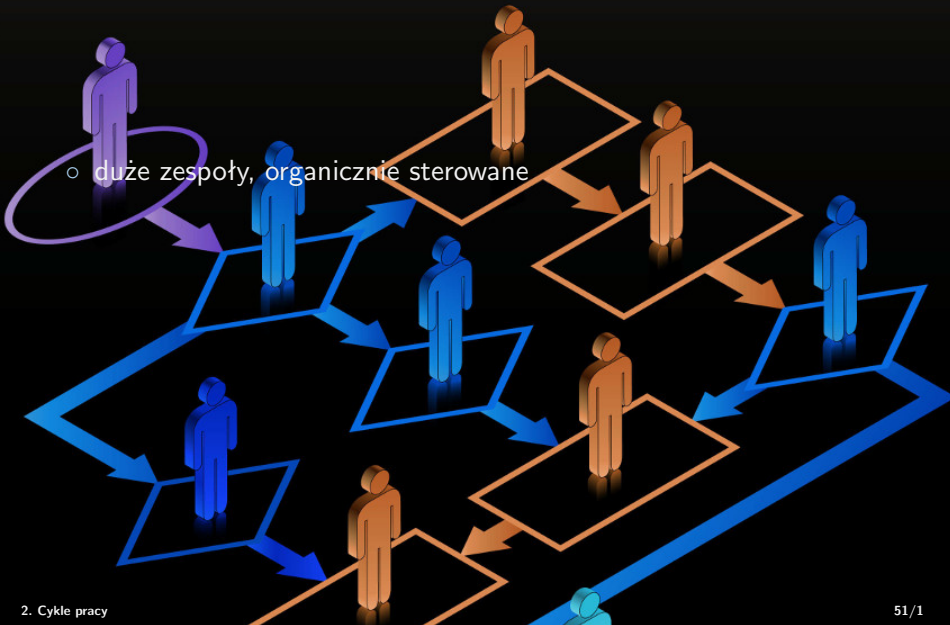
Forking workflow

- niescentralizowane
- każdy ma swój serwer zdalny
- każdy ma pełną władzę nad projektem (mogą się rozdzielać na różne implementacje)
- utrudnienia w komunikacji



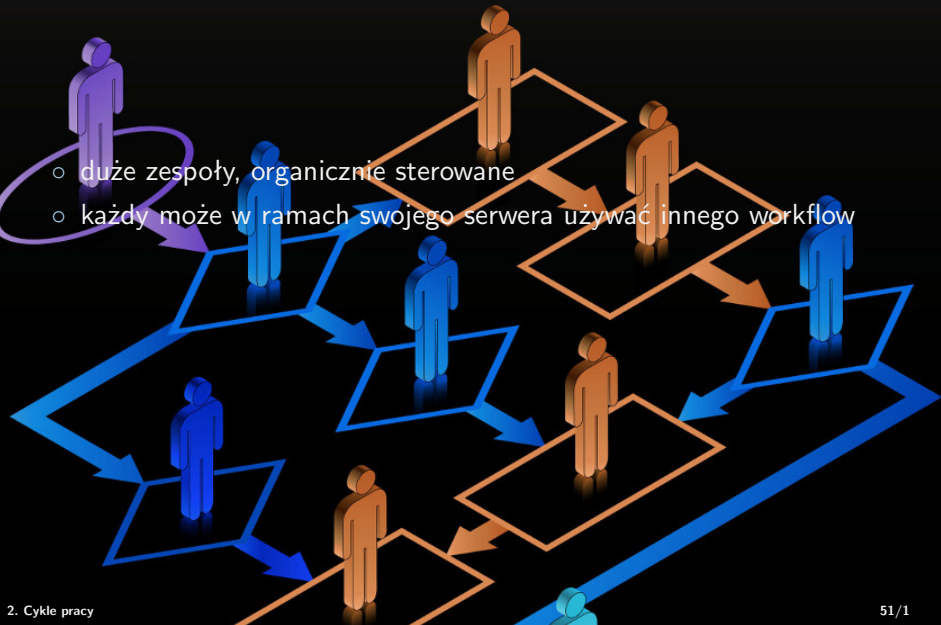
Forking workflow

- duże zespoły, organicznie sterowane



Forking workflow

- duże zespoły, organicznie sterowane
- każdy może w ramach swojego serwera używać innego workflow




Forking workflow

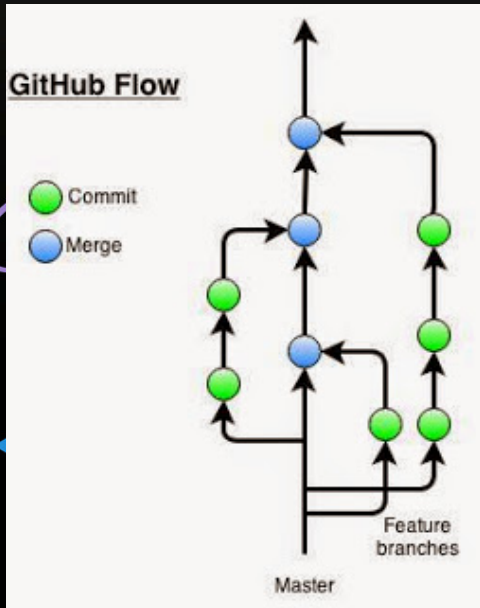
- duże zespoły, organicznie sterowane
- każdy może w ramach swojego serwera używać innego workflow
- firmy dają dostęp do oficjalnego repo tylko wybranym



Forking workflow

- 
- The diagram illustrates the Forking workflow. It features several stylized human figures in blue and orange, each standing on a rectangular platform. These platforms are interconnected by a network of blue and orange arrows, representing the flow of work or code. A purple oval highlights a specific figure in the top left, with a purple arrow pointing from it towards the center. The overall layout suggests a decentralized system where multiple independent paths or 'forks' exist, allowing different teams or individuals to work on their own versions of a project while maintaining a common origin.
- duże zespoły, organicznie sterowane
 - każdy może w ramach swojego serwera używać innego workflow
 - firmy dają dostęp do oficjalnego repo tylko wybranym
 - pull requesty bardzo wygodne

Github workflow



Github workflow

- o każdy commit mastera jest wersją oficjalną



Github workflow

- każdy commit mastera jest wersją oficjalną
- każda funkcjonalność to branch



Github workflow

- każdy commit mastera jest wersją oficjalną
- każda funkcjonalność to branch
- branch zakończony kończy się pull requestem




Github workflow

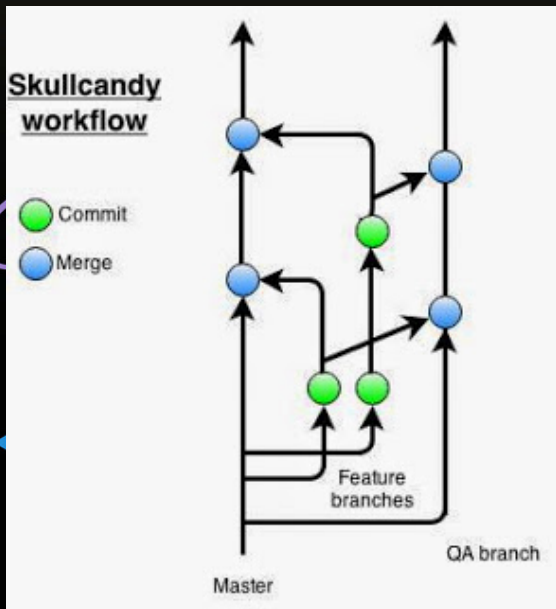
- każdy commit mastera jest wersją oficjalną
- każda funkcjonalność to branch
- branch zakończony kończy się pull requestem
- po zatwierdzeniu można zmergować się z masterem



Github workflow

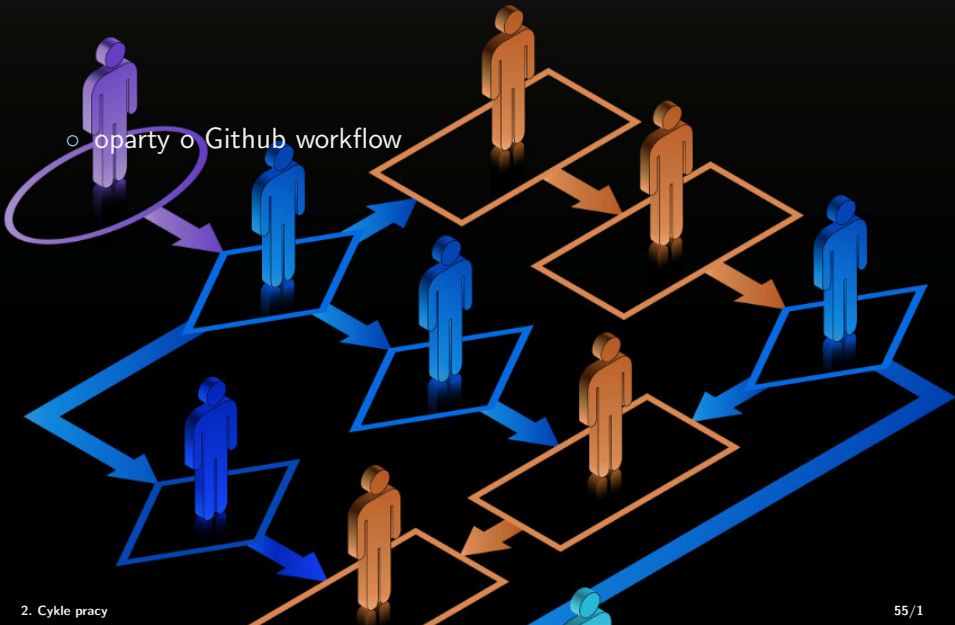
- 
- The diagram illustrates a structured GitHub workflow. It features several stylized human figures in blue and orange, each standing on a rectangular platform. These platforms are interconnected by a network of blue and orange arrows, representing the flow of code and development. A purple circle highlights a specific figure on the left, likely representing the master branch. The workflow shows how changes are made on feature branches, tested, and then merged back into the main branches (master or develop) via pull requests. The final step shows a merge back to master, which triggers a build process.
- każdy commit mastera jest wersją oficjalną
 - każda funkcjonalność to branch
 - branch zakończony kończy się pull requestem
 - po zatwierdzeniu można zmergeować się z masterem
 - po merge'u konieczne należy przesmoke'ować build

Skullcandy workflow



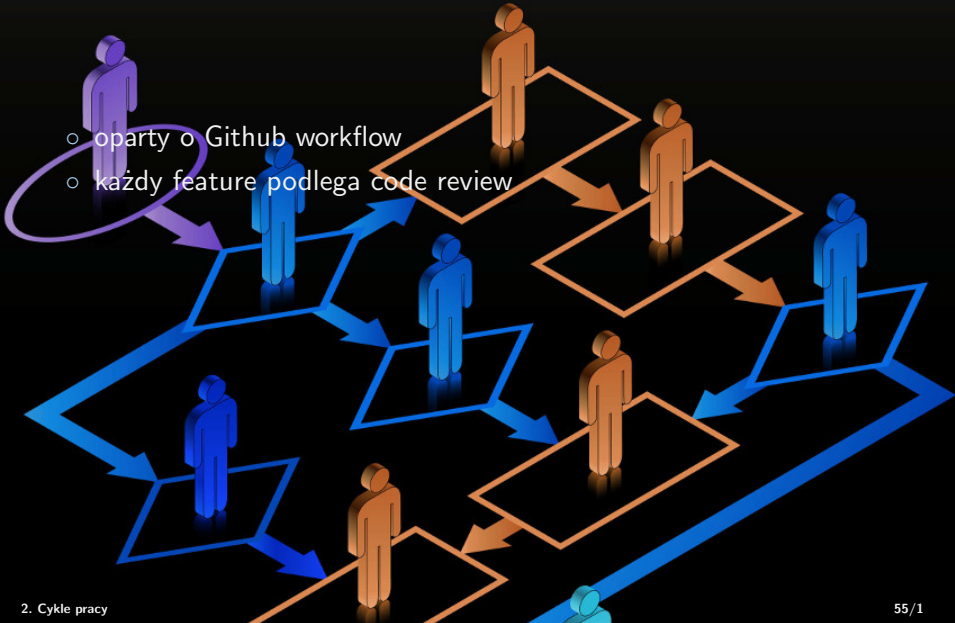
Skullcandy workflow

o oparty o Github workflow



Skullcandy workflow

- o oparty o Github workflow
- o każdy feature podlega code review



Skullcandy workflow

- o oparty o Github workflow
- o każdy feature podlega code review
- o feature jest najpierw merge'owany do gałęzi QA (dla testów), a dopiero zaakceptowana wersja wędruje do mastera



Skullcandy workflow

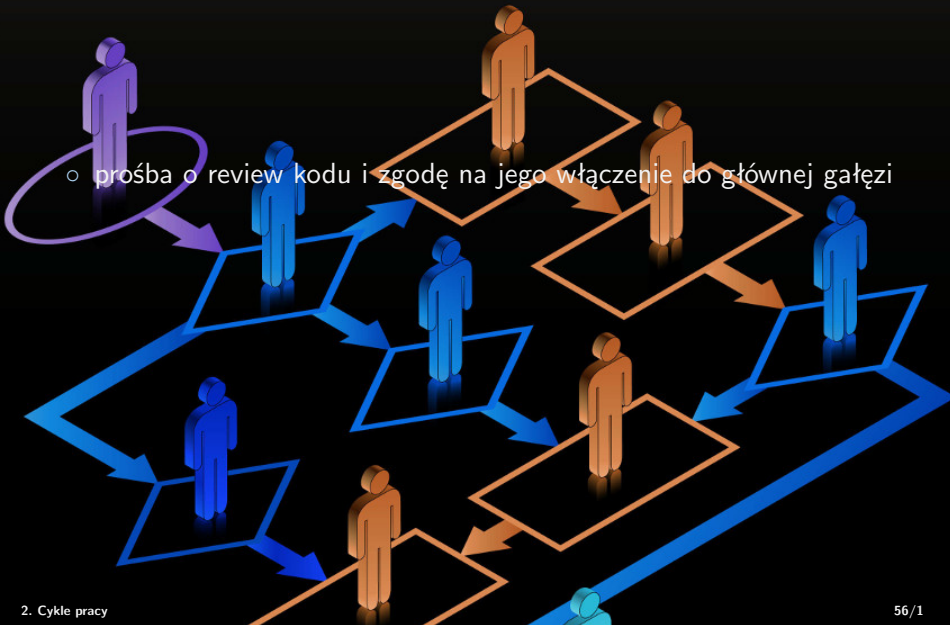
- o oparty o Github workflow
- o każdy feature podlega code review
- o feature jest najpierw merge'owany do gałęzi QA (dla testów), a dopiero zaakceptowana wersja wędruje do mastera
- o bazą wszystkiego jest master

Skullcandy workflow

- o oparty o Github workflow
- o każdy feature podlega code review
- o feature jest najpierw merge'owany do gałęzi QA (dla testów), a dopiero zaakceptowana wersja wędruje do mastera
- o bazą wszystkiego jest master
- o nazwy powiązane z funkcjonalnością, ale i z autorem

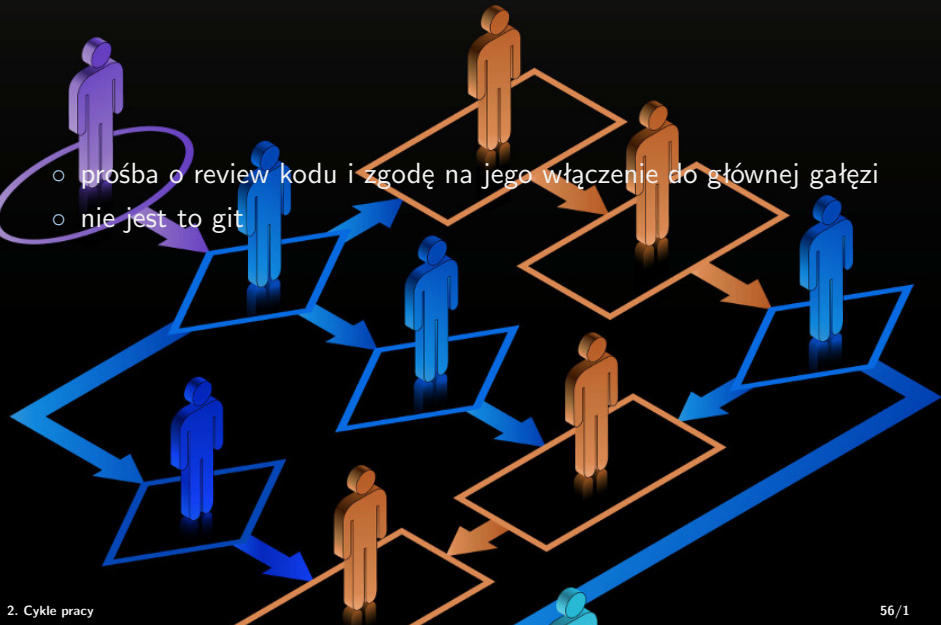
Pull request

- o prośba o review kodu i zgodę na jego włączenie do głównej gałęzi



Pull request

- prośba o review kodu i zgodę na jego włączenie do głównej gałęzi
- nie jest to git



Pull request

- prośba o review kodu i zgodę na jego włączenie do głównej gałęzi
- nie jest to git
- jest to platforma wymiany zdań na temat kodu, jak i umożliwiająca zatwierdzenie takiego kodu

Źródła

- 
1. <https://www.endpoint.com/blog/2014/05/02/git-workflows-that-work>
 2. <https://www.slideshare.net/noamkfir/git-workflows-40678584>
 3. <http://nvie.com/posts/a-successful-git-branching-model/>
 4. <https://code.tutsplus.com/tutorials/rewriting-history-with-git-rebase-cms-23191>