SZKOLENIE - GIT

USTALENIA:

- Cel i plan szkolenia
- Obowiązki bieżące
- Pytania, dyskusje, potrzeby
- Elastyczność zagadnień

PROWADZĄCY SZKOLENIE:

MATEUSZ KULESZA

- Senior Software Developer,
- Team Leader, Scrum Master
- Project Manager, Konsultant i szkoleniowiec
- Wieloletnie doświadczenie komercyjne w pracy z GIT

GIT

- Stworzony w 2005 roku przez Linusa Torvaldsa
- Najbardziej popularny system kontroli wersji na świecie
- Open Source
- Szybki, bezpieczny i rozproszony
- Może być początkowo nieintuicyjny

INSTALACJA I KONFIGURACJA

https://git-scm.com/download/win

JAK DZIAŁA GIT?

- Distributed Version Control System
- Każdy developer posiada kopię całej historii
- Możliwość pracy na wielu równoległych wersjach historii
- Możliwość sychronizacji zmian pomiędzy repozytoriami
- 3 drzewa - katalog roboczy, index i historia

GIT A SVN

- Commit w GIT jest lokalny, nie wysyła zmian
- Historia zmian przechowywana jest lokalnie
- Git pozwala przechować wiele równoległych historii
- Git pozwala na szybkie przełączanie pomiędzy historiami
- Przechowywanie całych plików zamiast samych zmian
- Historie zmian można wypchnąć na zdalny serwer

PRACA LOKALNA

Inicjalizacja i konfiguracja repozytorium

- `git init <directory>`

DODAWANIE PLIKÓW

- `git add <file>`
- `git add <directory>`
- `git add -p`

PIERWSZY COMMIT

- `git add .`
- `git commit`

GIT CONFIG

- `git config --local user.name "User Name"`
- `git config --global user.email "user@domain.com"`
- `.gitconfig`

GIT ALIAS

- `git config --global alias.ci commit`
- `git config --global alias.co checkout`
- `git config --global alias.br branch`
- `git config --global alias.ci commit`
- `git config --global alias.st status`

POPRAWIAMY OSTATNI COMMIT

- `git commit --amend`

PRACA Z INDEKSEM

- `git status`

JAK DZIAŁA GIT - INTERNALS

- `git hash-object -w test.txt`
- `find .git/objects -type f`
- `git cat-file -p <hash>`
- `git update-index --add --cacheinfo 100644 <hash> test.txt`
- `git write-tree`
- `git read-tree <tree-hash>`
- `git commit-tree <tree-hash>`

USUWANIE I PRZENOSZENIE PLIKÓW

- `git rm <file>`
- `git mv <source> <destination>`

IGNOROWANIE PLIKÓW

- `.gitignore`
- `.gitkeep`

PRZEGLĄDANIE HISTORII - LOG

- `git show ca3rqr`
- `git log -4`
- `git log --author="<pattern>"`
- `git log --grep="<pattern>"`
- `git log <since>..<until>`
- `git log <file>`
- `git log --graph --decorate --oneline`

WYŚWIETLANIE ZMIAN - DIFF

Porównanie z HEAD

- `git diff`

Porównanie z plikami w indeksie:

- `git diff --cached`

Porównanie między commitami:

- `git diff <commit> <commit>`
- `git diff <branch> <branch> <file>`

PRZYWRACANIE PLIKÓW - CHECKOUT

- `git checkout <commitOrRef> <file>`

COFANIE ZMIAN - REVERT

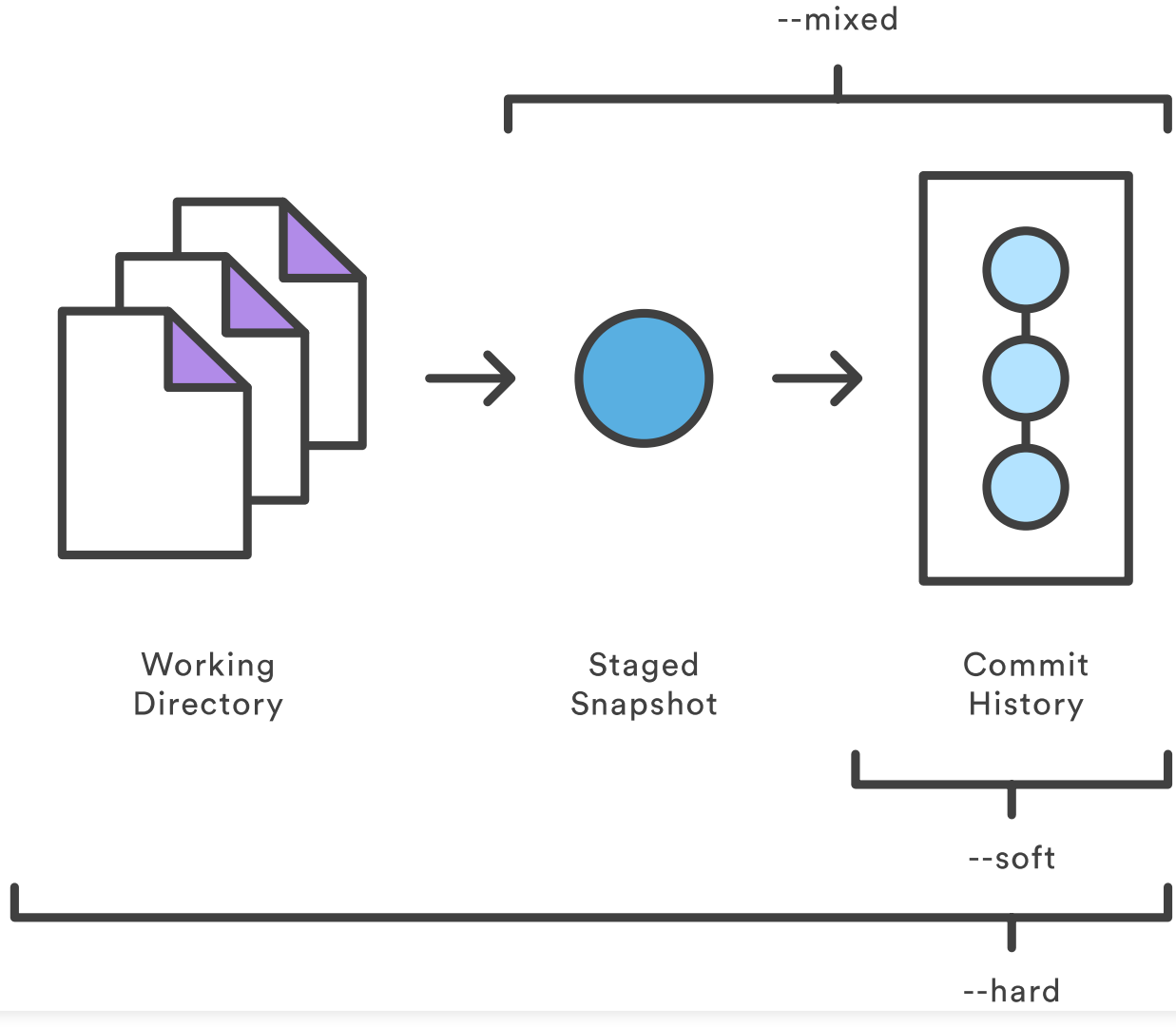
- `git revert <commit>`

EDYCJA HISTORII ZMIAN - RESET

- `git reset <file>`

TRYBY RESET - SOFT, MIXED, HARD

The scope of git reset's modes



ZAGUBIONE ZMIANY - REFLOG

- `git reflog`

PRZECHOWALNIA (STASH)

- `git stash`
- `git stash --include-untracked`
- `git stash show`
- `git stash list`
- `git stash pop stash@[2]`
- `git stash apply`
- `git stash branch add-stylesheet stash@{1}`
- `git stash drop stash@{1}`
- `git stash clear`

INTERAKTYWNA EDYCJA HISTORII - REBASE

```
git rebase -i
```

PRACA RÓWNOLEGŁA - GAŁĘZIE

ETYKIETY - TAG

- `git tag`
- `git tag -a v1.4 -m "my version 1.4"`

RÓWNOLEGŁE HISTORIE - BRANCH

- `git branch <nazwa-brancha>`
- `git checkout -b <nazwa-brancha>`

DETACHED HEAD

- `git checkout <commit>`

WYBIERANIE ZMIAN - CHERRY-PICK

- `git chery-pick`

ŁĄCZENIE HISTORII - MERGE

- `git merge`

ROZWIĄZYWANIE KONFLIKTÓW

- `git merge --resolve`
- `git merge --abort`

ŁĄCZENIE Z PRZESUNIĘCIEM - MERGE REBASE

- `git merge --rebase`

KLONOWANIE PROJEKTU - CLONE

```
git clone <source>
```

GITHUB - PIERWSZE REPOZYTORIUM

GITHUB - KLUCZE SSH

```
poprzednio
    repozytorium
    wyslalismy zmiany
    zalogowalismy sie z windows
logowanie
    nie jest bezpieczne
    trudno zmienic
    hasla przesylane w sieci
zmiany
    2 autorów
        Eduweb GIT
        Mateusz Kulesza
usuwamy poświadczenia
```

KONFIGURUJEMY KLUCZE

- `ssh-keygen`

https://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key

GAŁĘZIE ZDALNE - REMOTE

```
- `git remote show`
- `git remeote add`
    -f   - auto fetch
    --tags
    --no-tags
    -m - master / remote head
add rename remove
```

PRZESYŁANIE ZMIAN - PUSH

- git branch / git checkout -b --track origin/master
- git branch -u origin/master
- git branch --set-upstream origin/master
- git push -u origin <branch>
- git push --all -u
- git push --prune
  - --dry-run
  - --delete
- branch.master.remote=origin branch.master.merge=refs/heads/master
- git push origin --delete <branch_name>
- origin i push url

POBIERANIE ZMIAN - PULL I FETCH

Tylko pobierz

- `git fetch`

Pobierze i zmergeuj

- `git pull`

POBIERANIE ZE ZMIANĄ BAZY - PULL REBASE

- `git checkout -b praca_rownolegla -t origin/master`
- `git pull --rebase`