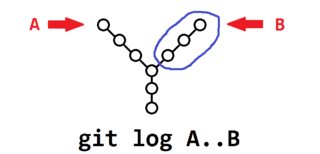
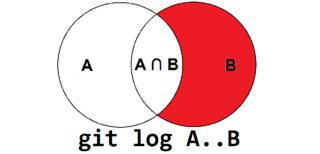
Popularne przełączniki do **git log**

git log --after="2014-02-12T16:36:00-07:00"  
git log --before="2014-02-12T16:36:00-07:00"  
git log --since="2 weeks 3 days 2 hours 30 minutes 59 seconds ago"  
*Wyszukiwanie commitów po czasie*  
git log -4 *Określanie liczby commitów (w tym przypadku 4 commity)*  
git log --oneline  
*Skompresowanie commitów jedna linijka per commit*  
git log --graph  
*Wizualizacja commitów w grafie*  
  
git log --decorate  
*Dodanie do commitów adnotacji o branchach/tagach itp.*   
  
git log --stat  
*Wylistowanie zmodyfikowanych/dodanych/usuniętych plików*  
git log --patch  
*Ukazanie diffów – zmian wprowdzonych przez dany commit*  
  
git log origin/develop --not develop  
*Commity z brancha origin/develop(„serwera”), których nie ma na branchu develop(„lokalnie”)*  
  
git log develop..origin/develop  
*Commity z brancha origin/develop(„serwera”), których nie ma na branchu develop(„lokalnie”)*  
  
git log develop...origin/develop  
*Commity z brancha origin/develop(„serwera”), których nie ma na branchu develop(„lokalnie”),  
oraz commity z brancha develop(„lokalnie”), których nie ma branch origin/develop(„serwer”)*  
  
git log --author="Toma\*"  
*Commity, których autor treści pasuje do regexa „Toma\*”*   
  
git log --committer="Toma\*"  
*Commity, które zostały stworzone przez osobę pasującą do regexa „Toma\*”*  
  
git log --grep=".scope\*"  
*Commity których wiadomość pasuje do regexa „.scope\*”*  
  
git log --no-merges  
*Wykluczneie commitów mergujących z wyników*  
  
git log –all  
*Commity z wszystkich branchy (nie tylko z „obecnego”)*

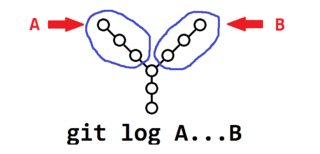
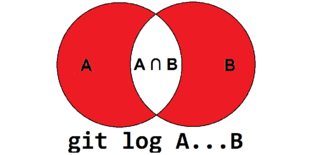
git log --pretty="FORMAT"  
*Zmiana wyświetlanego formatu commita, np. git log --pretty=”%an-%H”*https://devhints.io/git-log-format  
  
git log -S"com.arellomobile.mvp.MvpPresentert"  
*Committy które dodają/usuwają frazę „com.arellomobile.mvp.MvpPresentert”, tak zwany pickaxe*

## Visualization with Venn Diagrams & Commit Trees

Here is a visual representation of git log A..B. The commits that branch B contains that don't exist in A is what is returned by the commit range, and is highlighted in red in the Venn diagram, and circled in blue in the commit tree:



These are the diagrams for git log A...B. Notice that the commits that are **shared** by both branches are not returned by the command:



## Making the Triple-Dot Commit Range ... More Useful

You can make the triple-dot commit range ... more useful in a log command by using the --left-right option to show which commits belong to which branch:

$ git log --oneline --decorate --left-right --graph master...origin/master

< 1794bee (HEAD, master) Derp some more

> 6e6ce69 (origin/master, origin/HEAD) Add hello.txt

In the above output, you'll see the commits that belong to master are prefixed with <, while commits that belong to origin/master are prefixed with >.

*From: [https://stackoverflow.com/questions/462974/what-are-the-differences-between-double-dot-and-triple-dot-in-git-com#24186641](https://stackoverflow.com/questions/462974/what-are-the-differences-between-double-dot-and-triple-dot-in-git-com" \l "24186641)*

Wyjście ze strumienia logów za pomocą klawisza *q* :)