BALTIC TALENTS ACADEMY

GIT

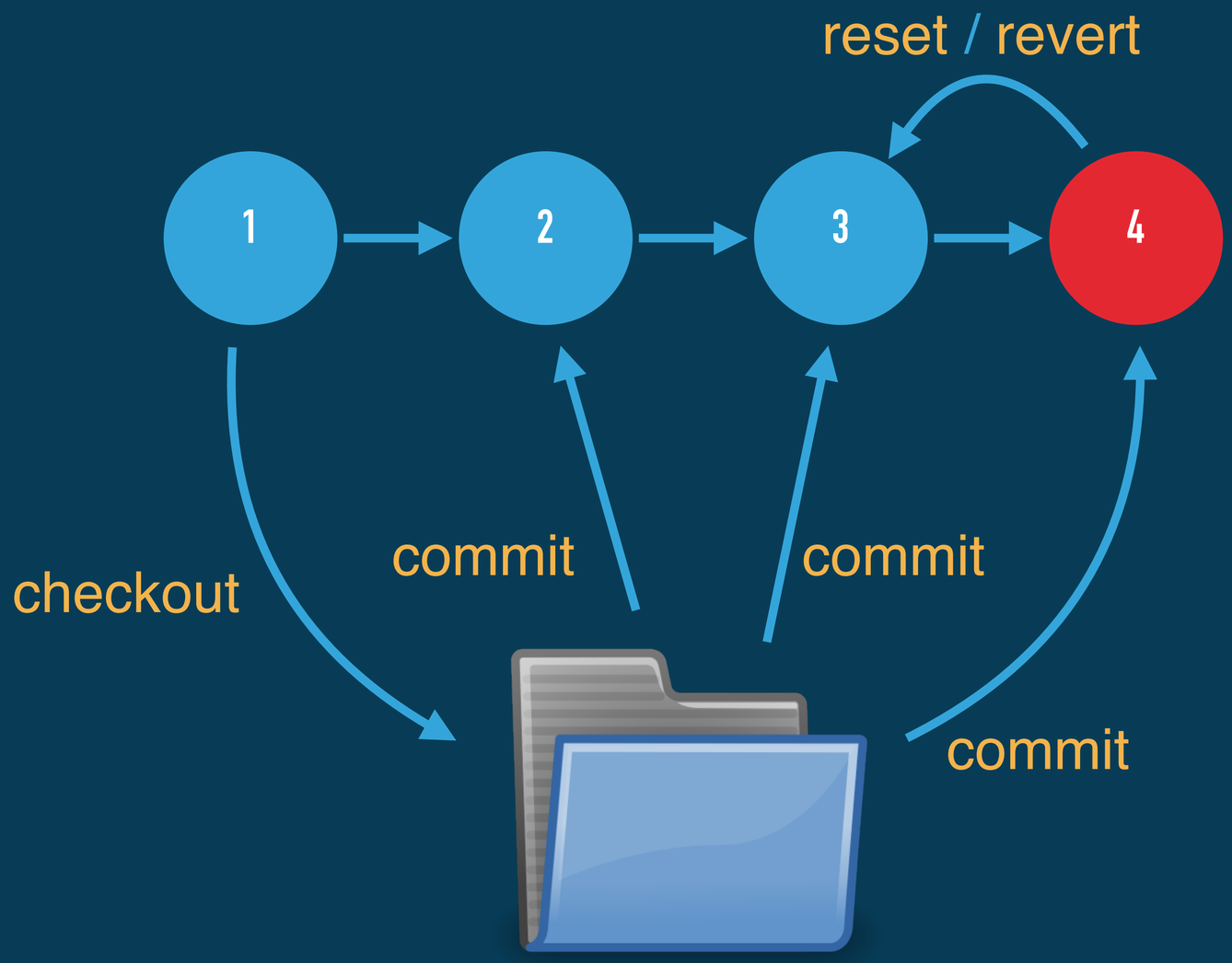
KAS YRA GIT

- Tai programų arba tiesiog bet kokių tekstų versijų kontrolės sistema (version control system - VCS)
- Sukurta 2005 Linus Torvalds, kad padėtų didelėms programuotojų grupėms dirbti kartu kuriant ir toliau tobulinant Linux operacinės sistemos branduolį
- Už GIT sistemos tolimesnį vystymą atsakingas Junio Hamano - https://git-blame.blogspot.com/
- GIT - a stupid content tracker

GIT SAUGYKLA (REPOSITORY)

- Tarkime mes turime kažkokį projektą
- Visi šitam projektui priklausantys failai (programos tekstai, moduliai, dokumentacija), jų pakeitimų istorija ir versijos sudaro visumą, kuri git sistemoje yra vadinama “saugykla (repository)”
- Git susideda bent jau iš lokalios (local repository) saugyklos esančios git naudotojo kompiuteryje, o taip pat ji gali ir turėti saugyklą kažkur kitame serveryje vadinama nutolusia saugykla (remote repository) - todėl apie git sakoma, kad tai paskirstyta (distributed) sistema

VERSIJOS



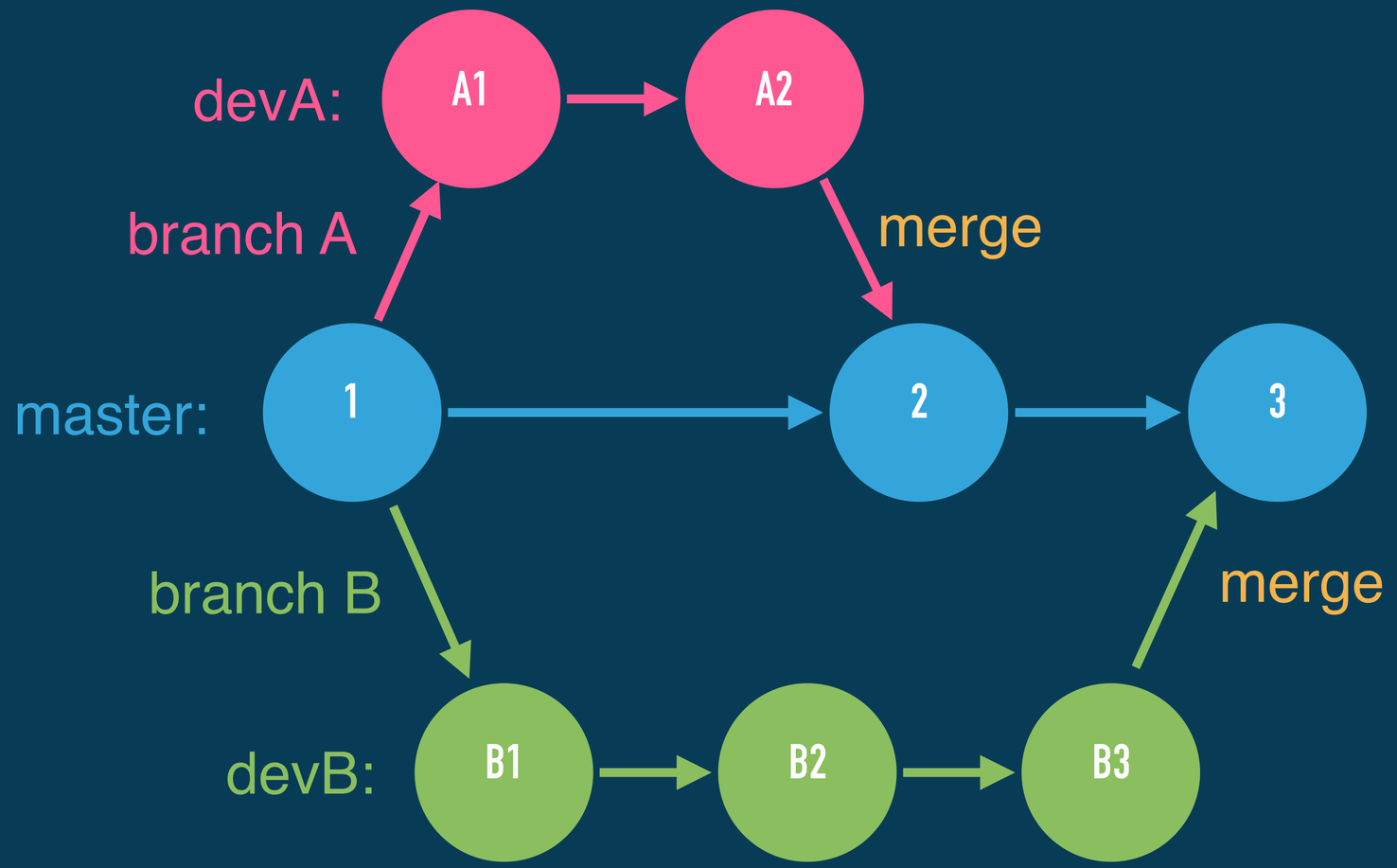
GIT ATŠAKOS (BRANCH)

- Atšakos (branches) - tai galimybė dirbti kartu keliems programuotojams vienu metu. Arba tam pačiam programuotojui dirbti vienu metu prie kelių skirtingų uždavinių
- Saugykla pagal nutylėjimą visada turi bent vieną atšaką vardu master
- Master atšaka sukuriama git saugyklos sukūrimo metu
- Paprastai master atšakoje laikomas ta kodo versija, kuri dabar realiai naudojama (ne testinė)

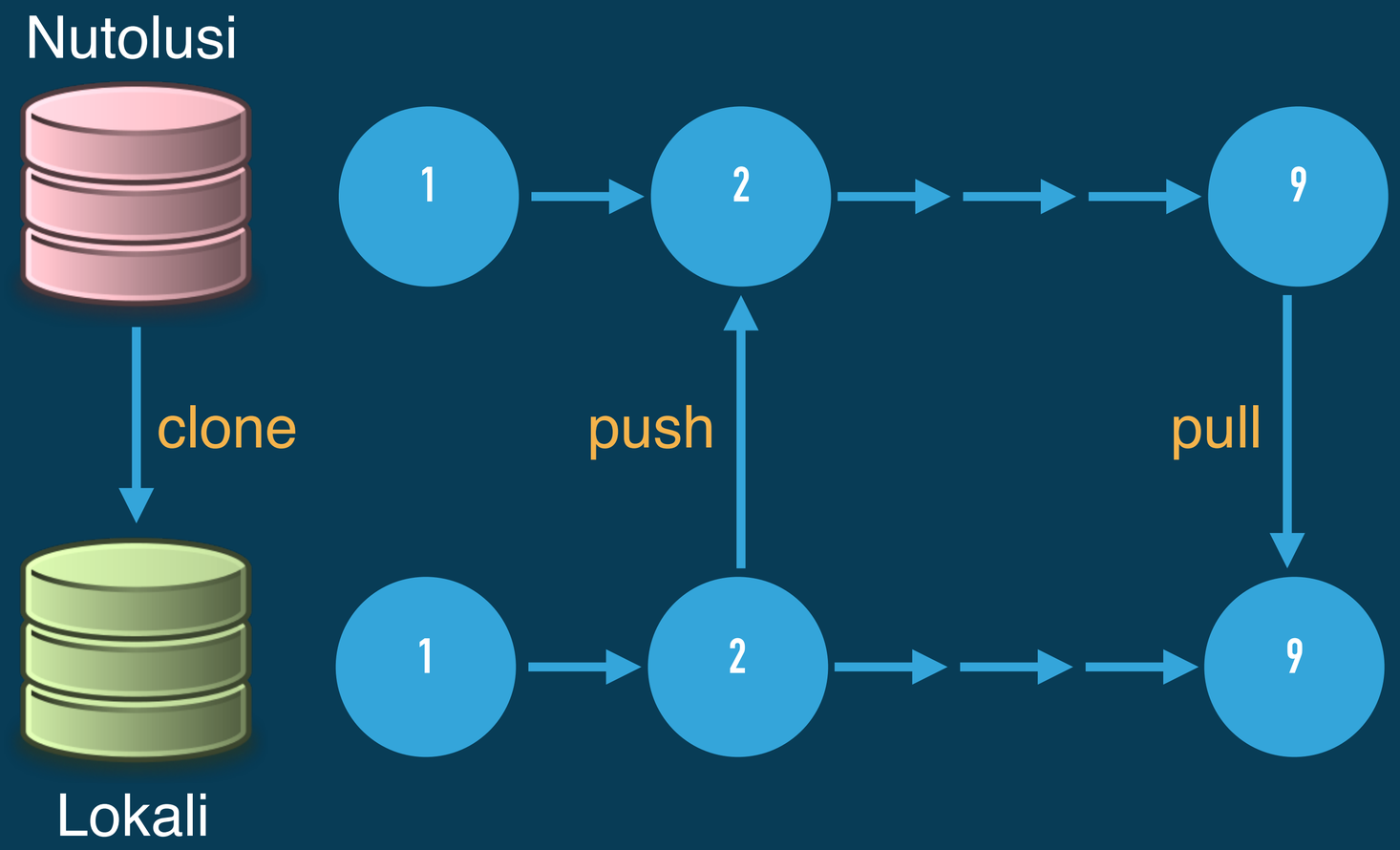
GIT ATŠAKOS (BRANCH) - TĘSINYS

- Atšakos naudojamas tada kai norime dirbti prie kažkokių patobulinimų:
  - Kuriame naują atšaką nuo **master** (ar kitos) atšakos
  - Persijungiame į tą atšaką ir ten atliekame savo darbus
  - Kai viskas atlikta ir viskas dirba be klaidų tai suliejame savo atšaką su **master** (ar kita) atšaka

ATŠAKOS


devA:
A1
A2
branch A
merge
master:
1
2
3
branch B
merge
devB:
B1
B2
B3

SAUGYKLOS (REPOSITORIES)



FAILŲ BŪSENA

- Kiekvienas git lokalios saugyklos failas gali būti vienoje iš trijų būsenų:
  - užfiksuota (commited) - failo pakeitimai jau įrašyti į git saugyklą
  - modifikuota (modified) - failas modifikuotas bet dar neįrašytas į saugyklą. Jei failas nebus specialiai pažymėtas (staged), jis ir nebus įrašytas į saugyklą
  - ruošiamas (staged) - failas modifikuotas ir dar neįrašytas į saugyklą, bet pažymėtas, kad bus įrašytas į saugyklą su sekančia commit komanda

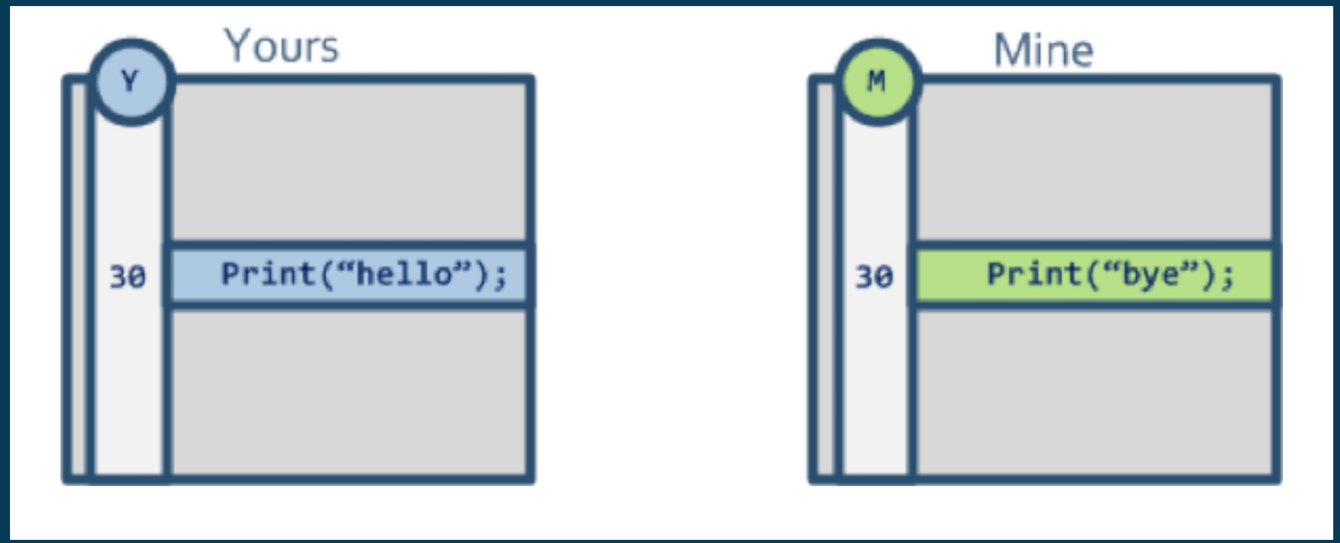
p.s. jei mums reikia persijungti į kitą atšaką arba ištraukti atšakos failus iš nutolusios saugyklos tai reikia, kad visi mūsų lokalūs failai būtų commited būsenoje

GIT DARBO PRINCIPAI

- Kai persijungiame į reikiamą atšaką ir atkrauname jos failus, tai patenkame į taip vadinamą darbinę direktoriją ar medį (working directory/tree/area)
- Kai sukuriame naują failą, tai norint kad jis patektų į saugyklą, reikia jį pridėti į specialų sąrašą (staging area/index), t.y. į tokį sąrašą, iš kurio git sistema žino kokių failų pakeitimus reikia saugoti su patvirtinimo (commit) komanda.
- Visi jau egzistuojantys saugykloje failų taisymai automatiškai papuola į tą sąrašą
- Mes galime pasirinkti kokių ir kokie failų pakeitimai papuls į lokalią saugyklą su sekančia saugojimo komanda

2-WAY MERGE

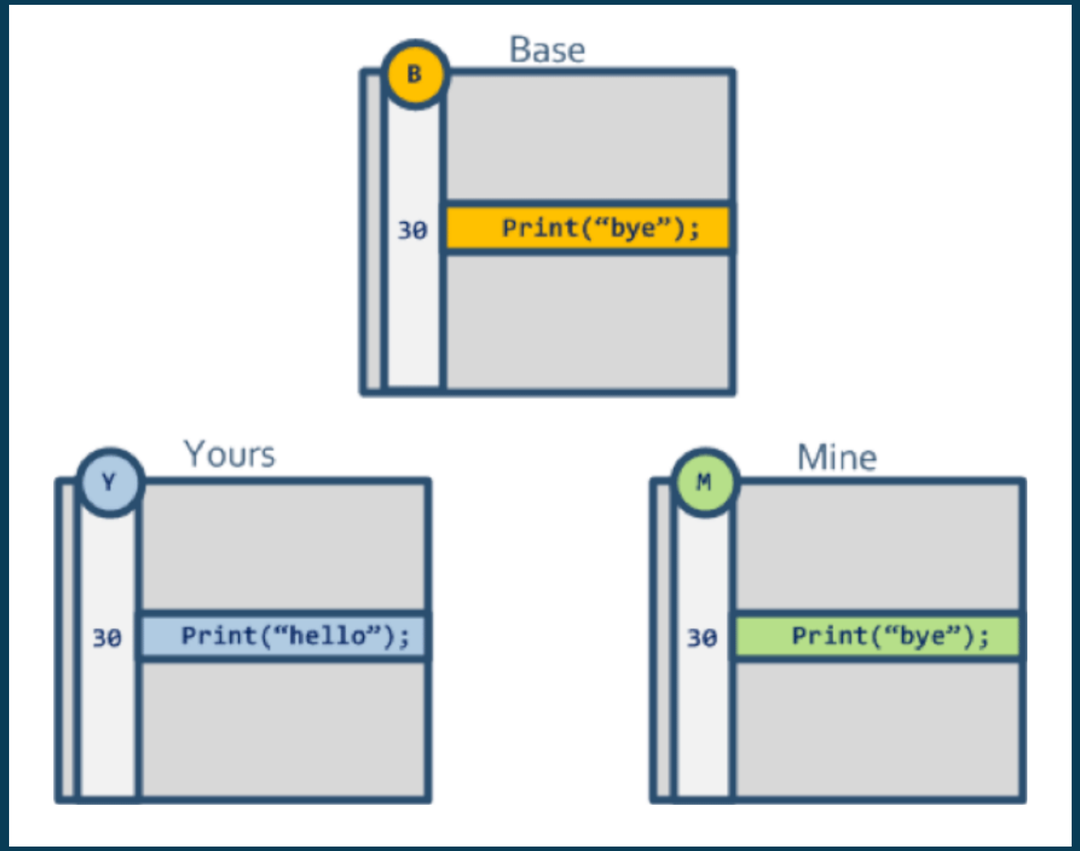
- Tarkime **aš** ir kažkas **kitas** modifikuojame tą patį failą (jo kopiją) vienu metu
- Ir dabar kažkas trečias turi abi failo kopijas ir mato, kad failai skiriasi 30 eilutėje



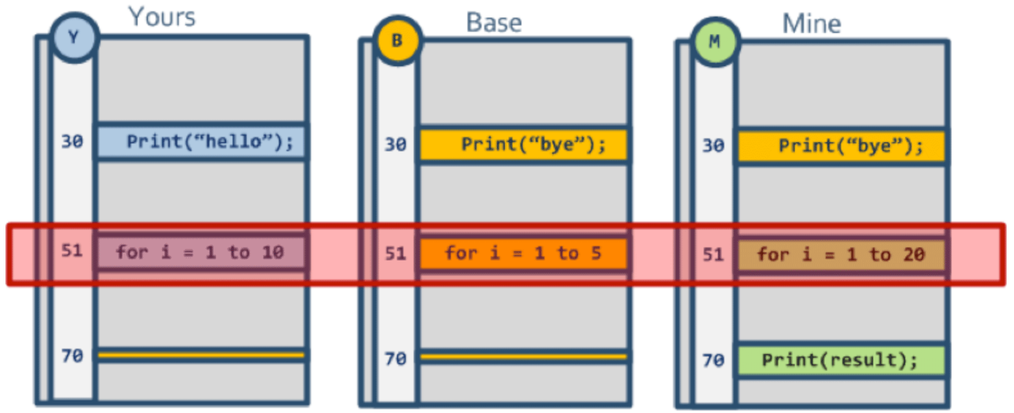
- Kaip žinoti kas pakeitė 30 eilutę - **aš** ar **kitas**?
- O gal mes abu keitėme tą eilutę?

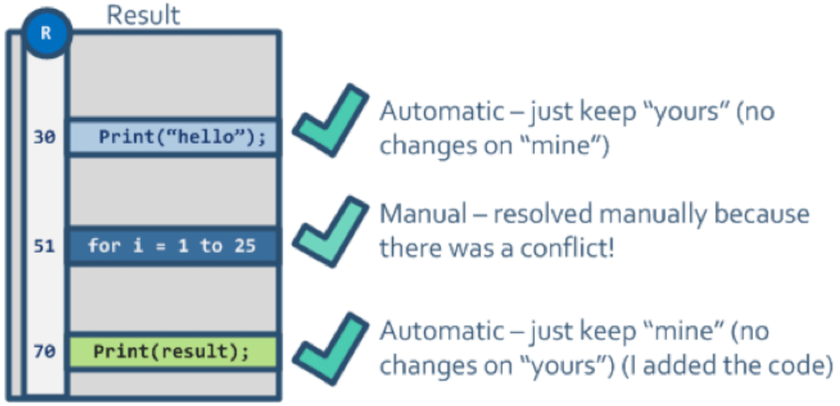
3-WAY MERGE

- Viskas pasidaro žymiai paprasčiau, jei turime pradinį failą (base)
- Dabar trečiasis aiškiai mato, kad pakeitimą padarė **kitas** ir gali automatiškai sulieti
- Tas bazinis (base) failas yra pats artimiausias bendras protėvis mano ir kito failui



KONFLIKTAI


Yours
Y
30
Print("hello");
51
for i = 1 to 10
70
Base
B
30
Print("bye");
51
for i = 1 to 5
70
Mine
M
30
Print("bye");
51
for i = 1 to 20
70
Print(result);


Result
R
30
Print("hello");
51
for i = 1 to 25
70
Print(result);
Automatic – just keep "yours" (no changes on "mine")
Manual – resolved manually because there was a conflict!
Automatic – just keep "mine" (no changes on "yours") (I added the code)

GIT TAG

- Kiekvienas git commit turi tam tikrą vardą arba id, bet pagal jį nėra patogu orientuotis
- Tam tikslui galima naudoti taip vadinamus git tag’us
- Tokie užtaginti commit’ai žymiai lengviau randami ir ypač naudingi pažymint naujos versijos ar subversijos išleidimą

GIT DIEGIMAS

- Visą git sistemą galima įdiegti iš čia: https://git-scm.com/download/
- Kompiuteriuose su Mac OS X ar Linux git greičiausia jau bus įdiegtas
- Tame pat puslapyje yra nuoroda į programas padedančias dirbti su git grafinėje aplinkoje - pvz.: SourceTree https://www.sourcetreeapp.com/ ar GitHub Desktop https://desktop.github.com/
- Diegiame…

GIT KOMANDOS - VISUAL STUDIO CODE

1. Show Command Palette: Ctrl+Shift+P, F1 (⇧⌘P)
2. Ieškome ir vykdome Git: Clone
3. …

NUORODOS

1. https://github.com
2. https://bitbucket.org
3. https://backlog.com/git-tutorial/en/ 
4. https://git-scm.com/book/en/v2
5. https://www.atlassian.com/git/tutorials
6. http://www.drdobbs.com/tools/three-way-merging-a-look-under-the-hood/240164902
7. https://git-blame.blogspot.com

PRATIMAS / UŽDAVINYS

1. Įdiekite, jei reikia, git ir SourceTree ar GitHub Desktop savo kompiuteryje
2. Pasitikrinkite ar git susietas su Visual Studio Code ir/ar IntelliJ
3. Užsiregistruokite github’e
4. Sukurkite github’e savo repositorių
5. Klonuokite jį savo kompiuteryje (naudokite SourceTree, GitHub Desktop programas arba tiesiai iš redaktoriaus Visual Studio Code ir/ar IntelliJ)
6. Sukurkite failą ir atlikite atitinkamus veiksmus, kad jis atsirastų github’o saugykloje (repositoriuje)
7. Sukurkite kitą katalogą su jame kažkokiais tekstiniais failais. Sukurkite kitą github’e repositorių.
8. Pasinaudodami git iš komandinės eilutės (!) sukurkite lokalų repositorių ir užkraukite failus į github’ą
9. Modifikuokite failą ir vėl užkraukite naują versiją į github’ą
10. Sukurkite naują branch’ą ir jame naują failą ir užkraukite į github’ą.
11. Pasitikrinkite ar persijungus branch’us naujai sukurtas failas dingsta ir vėl atsiranda.