Разработка на софтуер

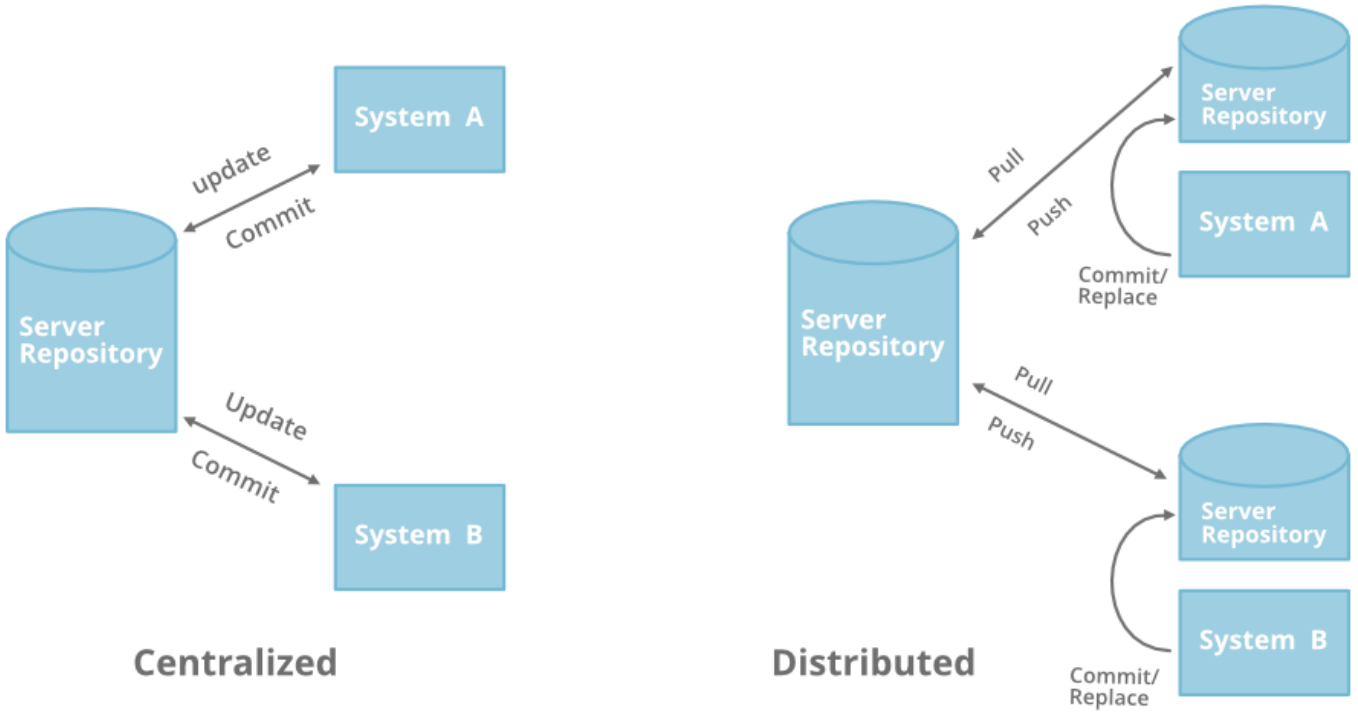
Лекция 2 – Version Control

Милен Спасов

Какво е version control система

- Система, която проследява и записва промените по даден файл или набор от файлове (repository)
- Поддържа пълна история на всички промени по всеки файл в проекта
- Улеснява интеграцията, създаването и управлението на софтуерни версии
- Видове version control системи
  - Централизирани
  - Дистрибутирани
- Примери:
  - Централизирани: CVS, Subversion
  - Дистрибутирани: Git, Mercurial, Bazaar

Centralized vs. Distributed version control

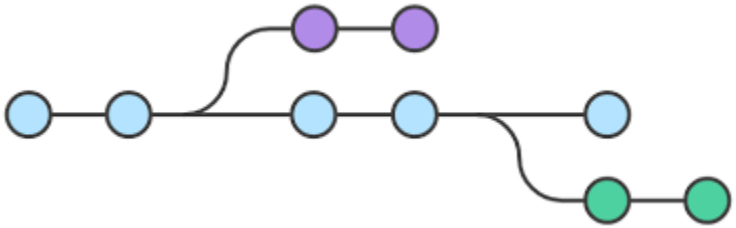


- Предимства на дистрибутираните системи:
  - Програмистите могат да работят без постоянна връзка помжду си
  - Множество пълни копия на проекта служат и като backup
  - Бързо и лесно merge-ване и разрешаване на конфликти

Git

- Git е безплатна open source дистрибутирана система, подходяща както за малки, така и за големи проекти и екипи
- В момента Git е практически без аналог и най-широко използваната система при стартиране на нови проекти
- Всяка локална Git директория съдържа пълно копие на repository-то с пълна история
- Няма централизиран сървър и не е необходима постоянна връзка между работещите по проекта

Какво е branch и основни branch-ове във всеки проект



- Branch-ът е разклонение на проекта, започващо от даден commit
- Commit е всяка промяна в проекта, която някой добавя в repository-то (напр. промяна във файл или добавен/изтрит файл)
- Branch-овете позволяват на много хора да работят в паралел по различни функционалности без да си пречат и в последствие да интегрират работата си с тази на останалите
- В един обичаен проект е препоръчително да има минимум следните основни бранчове:
  - Development
  - QA (или Testing)
  - Staging (или Pre-Production)
  - Production (или Master)

Среди за разработка (environments)

- Средите за разработка са напълно отделени една от друга, използват различни бази данни и версии на тях се качват по различно време

Development

- Използва се от програмистите за нови функционалности
- През повечето време нестабилна среда
- Не съдържа клиентски данни

Testing

- Използва се за тестване от QA екипа
- По-стабилна версия от development
- Не съдържа клиентски данни, само тестови

Staging

- Използва се от QA екипа и от клиенти за приемане на нови версии на проекта
- UAT (User Acceptance Testing) – тестове от крайния клиент
- Частично реални клиенски данни

Production

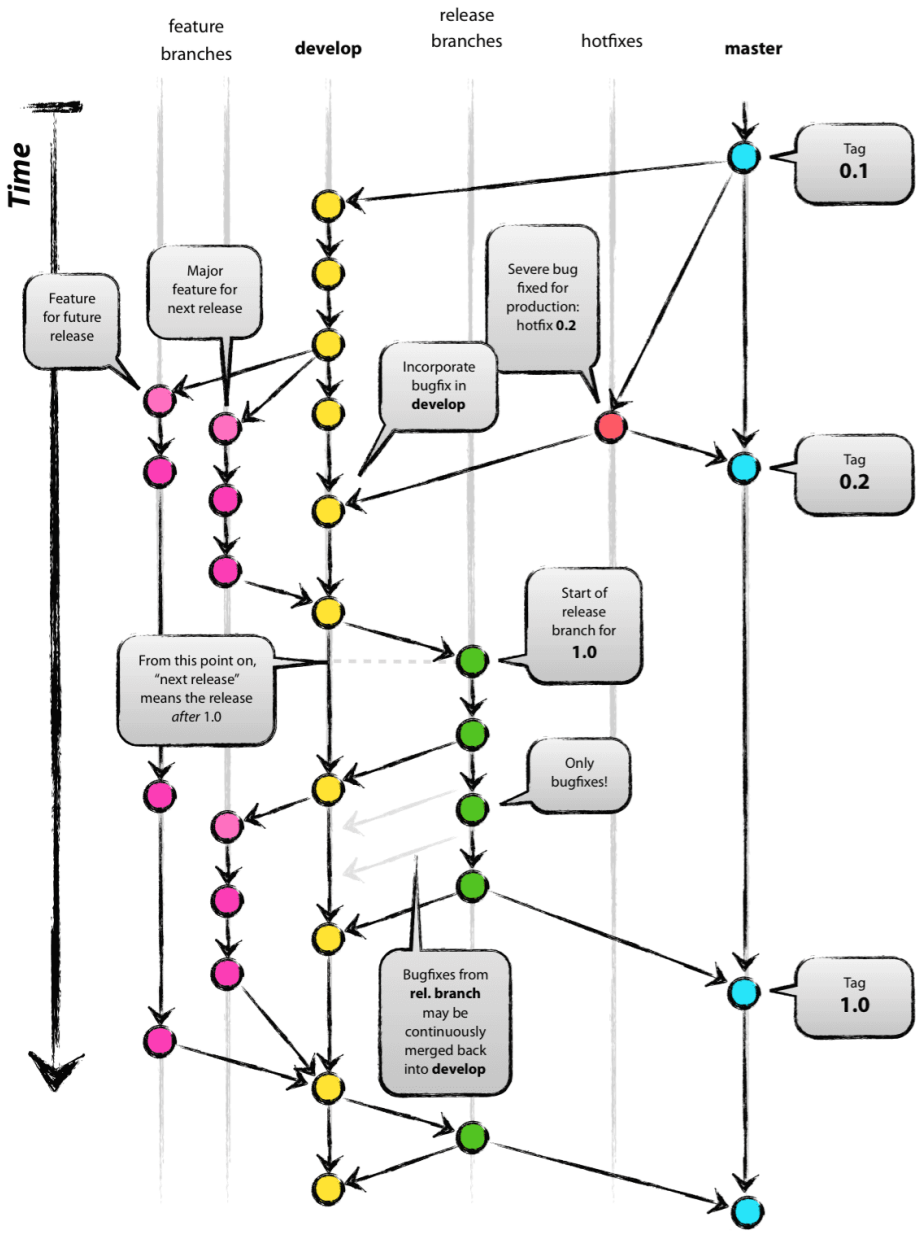
- Среда използвана активно от клиентите
- Съдържа всички клиентски данни
- Директна работа с тази среда и директно качване на версии на тази среда е недопустимо
- В много случаи програмистите нямат достъп до тази среда

Branching стратегия

При имплементация на нова промяна или функционалност обикновено се прилага следната последователност:

1. Програмистът взима текущия development branch
2. От него създава feature branch за необходимата промяна
3. Имплементира промените
4. Създава pull request към development branch-а
5. Друг програмист му прави код ревю
6. Промяната става част (merge) от development branch-а
7. Feature branch-а се изтрива

Директно качване в development branch-а без pull request не е добра практика.



Git команди (1)

- Проверка на статус на файл – засичане на промени по файлове, нови или премахнати файлове:
  - ***git status***

- Промяна на статуса на файл – добавяне на файл към проекта:
  - ***git add filename***

- Показване на разликите между файлове:
  - ***git diff***

- Премахване на файло от проекта:
  - ***git rm filename***

- Преименуване на файл:
  - ***git mv file_from file_to***

- Създаване на бранч в локалното repository:
  - ***git checkout -b branch_name***

- Изтриване на бранч от локалното repository:
  - ***git branch -d branch_name***

Git команди (2)

- Клониране на съществуващ проект:
  - ***git clone URL***

- Създаване на ново repository:
  - ***git init***
  - ***git add *.****
  - ***git commit -m "Initial commit"***
  - ***git push origin master***

- Commit на направени промени:
  - ***git commit***

- Commit с коментар (препоръчително):
  - ***git commit -m "Comment"***

- Преглеждане на commit историята:
  - ***git log***

- Преглеждане на разликите във всеки commit:
  - ***git log p***

Git команди (3)

- Промяна на commit без създаване на нов:
  - ***git commit -m "Comment"*** // Създаване на commit
  - ***git add filename*** // Забравен файл към commit
  - ***git commit -amend*** // Добавяне без създаване на нов commit

- Публикуване на нашия код в отдалеченото repository:
  - ***git push origin branch_name***

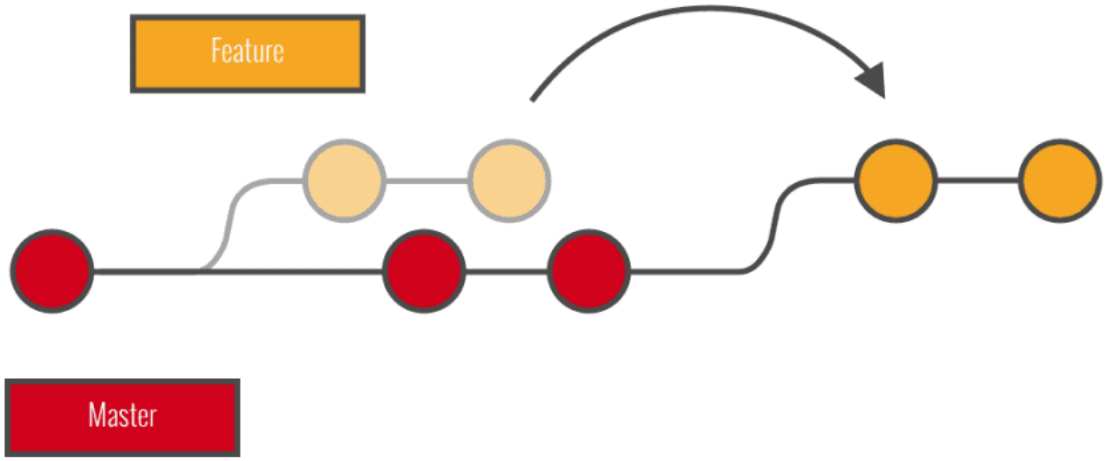
- Изтегляне на нови промени от отдалеченото repository:
  - ***git pull***

- Изтегляне на нови бранчове в локалната директория:
  - ***git fetch***

- Merge-ване на съществуващи паралелни промени, запазвайки историята на промените във всеки от тях:
  - ***git checkout dev*** // Сменяме към development бранча
  - ***git pull*** // Изтегляме последните промени от repository-то
  - ***git merge story_1145*** // Merge-ваме промените от наш локален бранч в development
  - ***git push origin dev*** // Качваме новия development бранч

Git команди (4)

- Rebasing - преместване на един (или повече) commit към нова базов commit:
  - ***git rebase branch_name***



Домашно

Допълнителна информация и материали за работа с git:

- https://guides.github.com/introduction/git-handbook/
- https://www.atlassian.com/git/tutorials
- https://www.youtube.com/watch?v=4XpnKHJAok8
- https://www.sourcetreeapp.com/


Thank
You
köszönöm
Kiitos
Vinaka
Maake
Asante
Shukria
Dhanyavadagalu
감사합니다
Dankscheen
спасибо
Kam Sah Hammida
شكرا
Manana
Dankon
Matondo
Dank Je
Mauruuru
Biyan
Diolch i Chi
Terima Kasih
Taiku
Tack
Blagodaram
Ngiyabonga
Dziekuje
Chokrane
Grazie
Juspaxar
Arigato
Mochchakkeram
Gracias
Khap
Tingki
நன்றி
Bedankt
D'akujem
धन्यवाद
cảm ơn bạn
Paldies
Gratias Tibi
Ua Tsaug Rau Koj
Grazas
Mèsi
Obrigado
Kia Ora
Kop Khun
ありがとう
Děkuji
Nirringrazzjak
Di Ou
Suksama
Matur Nuwun
Rahmat
谢谢
хвала
Hvala
Welalin
Djiere Dieuf
Misaotra
Danke
Merci
Salamat
Go Raibh Maith Agat
ขอขอบคุณคุณ
Najis Tuke
Eskerrik Asko