

СУЧАСНІ ТЕХНОЛОГІЇ ***ПРОГРАМУВАННЯ***

122 «Комп'ютерні науки»
КН-18

2019 / 2020 навчальний рік

Технології # 6.

Системи керування версіями

Колективна робота.

1. Системи керування версіями.
2. Git. Стан та контроль файлів.
3. Git. Знімки.
4. Git. Робота з гілками.

https://github.com/eabshkvprof/2020_Mod_Prog_Techn

VCS. Системи керування версіями

VCS (*Version Control System*) -

програмний інструмент для керування версіями програмної інформації: початкового коду програми, скрипту, веб-сторінки, веб-сайту, тексту ...

Два класи:

Централізовані – вся інформація зберігається в єдиному «серверному» сховищі. Обмін файлами відбувається через центральний сервер.

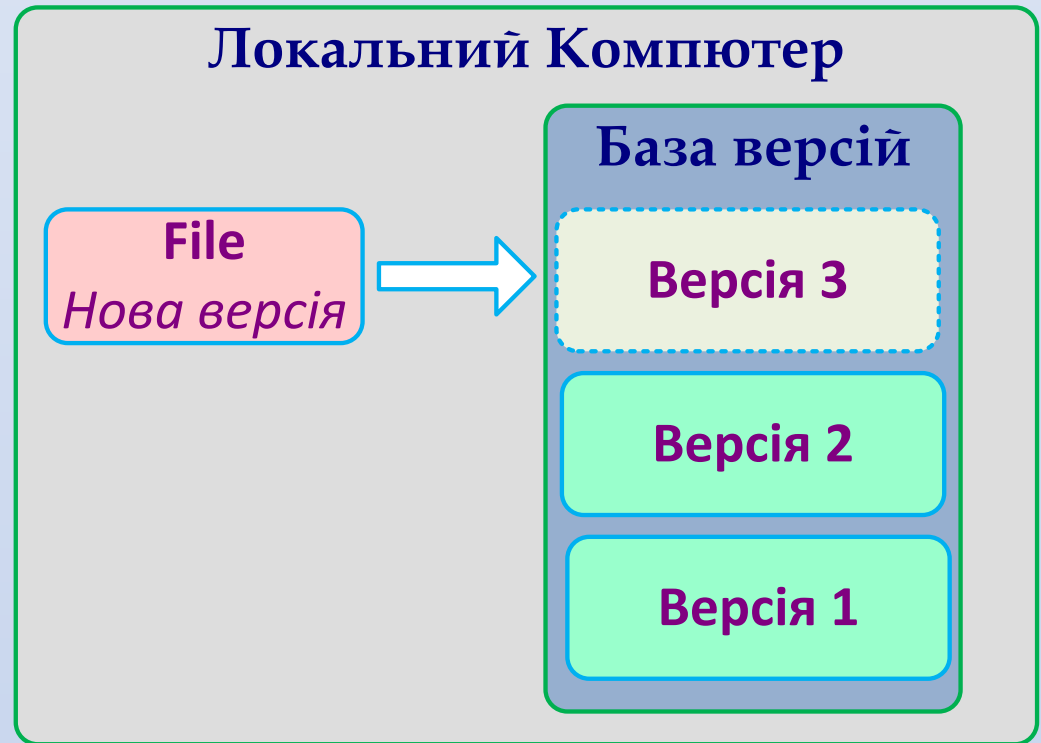
Підтримується можливість створення та роботи з локальними робочими копіями.

Розподілені – використовує розподілену модель зберігання файлів, не потребує сервера: всі файли знаходяться на кожному з робочих комп'ютерів розробників.

VCS. Локальна система

«Наївний» підхід -
копіювання файлів в
окрему директорію.

Найпростіша VCS -
**зберігає відмінності
між файлами** в
спеціальному форматі
на диску. Може
відтворити будь-який
файл в будь-який
момент часу.

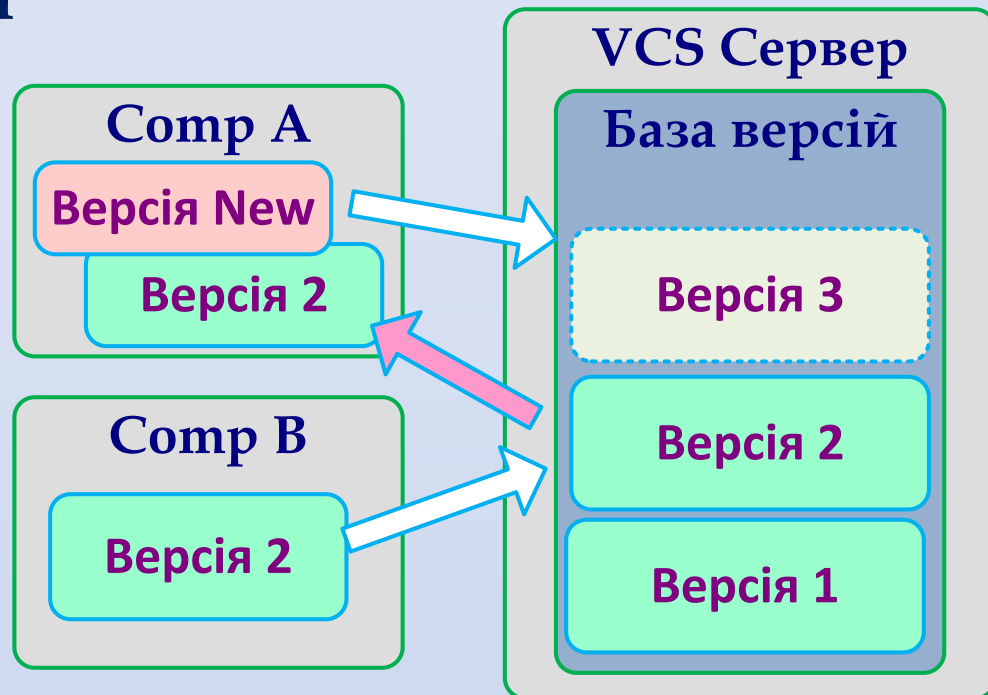


Revision Control System (RCS, 1985) - при зберіганні
текстових файлів використовується алгоритм дельта-
компресії, коли зберігається тільки перша версія і всі
наступні зміни.

VCS. Централізована система

! Колективна розробка

Система має єдиний сервер, який містить всі версії файлів. Клієнти (розробники) отримують файли з центральної бази (**репозиторію**).



Декілька клієнтів працюють одночасно. Коли зміни приймаються номери версій змінених файлів автоматично збільшуються, і сервер записує коментар, дату і ім'я користувача в свій журнал.

CVS (Concurrent Versions System), 1990 – 2008

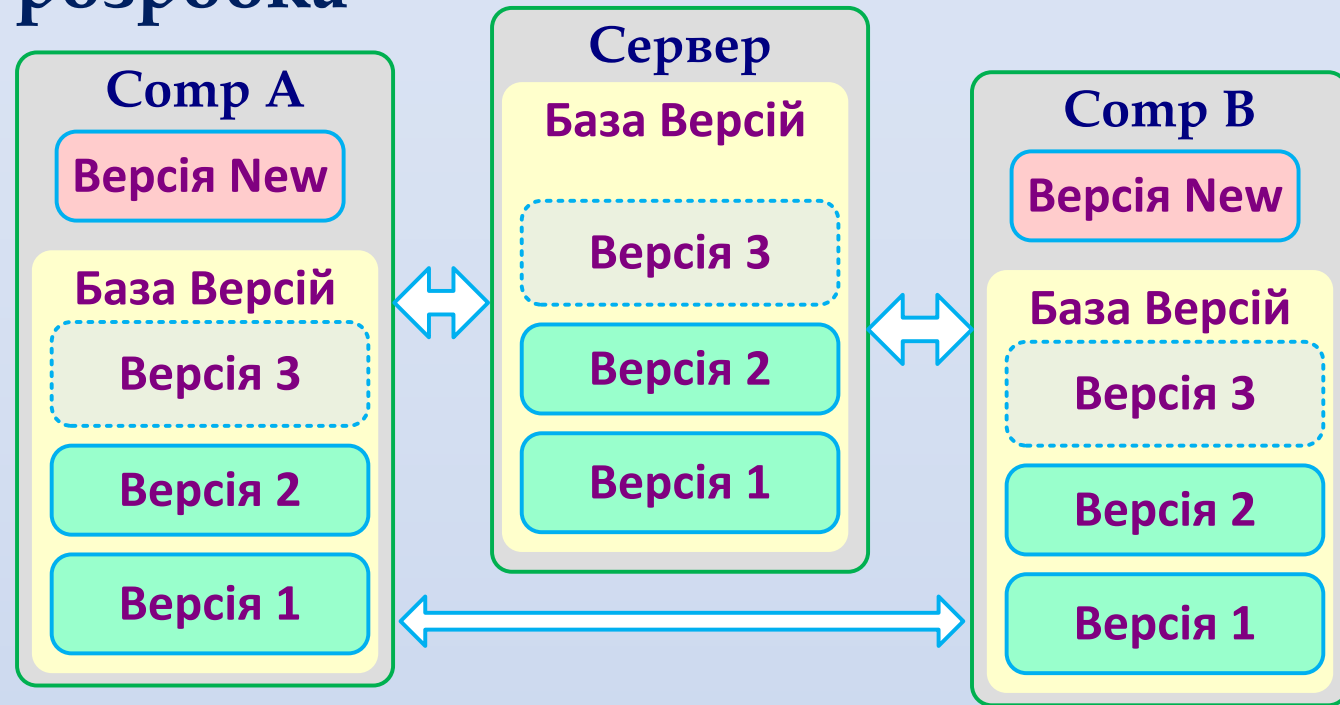
Perforce (P4), 1995 - ?

Subversion (SVN), 2004 →

VCS. Децентралізована система

! Колективна розробка

Локальні комп'ютери мають **повну** копію репозиторію



Системи підтримують взаємодію з декількома віддаленими репозиторіями, що дозволяє співпрацювати з різними групами розробників і мати одночасно декілька типів робочих процесів.

Git, 2005 --
Bazaar, 2005 – 2016

.....

Git

Розподілена система керування версіями файлів та спільної роботи. Розробник L.Torvalds (Linux).

Git – надійна, високоефективна децентралізована VCS, надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок (тисячі паралельних гілок).

Цілісність історії та стійкість до змін забезпечується криптографічними методами, можлива прив'язка цифрових підписів.

<https://en.wikipedia.org/wiki/Git>

Git

VCS –

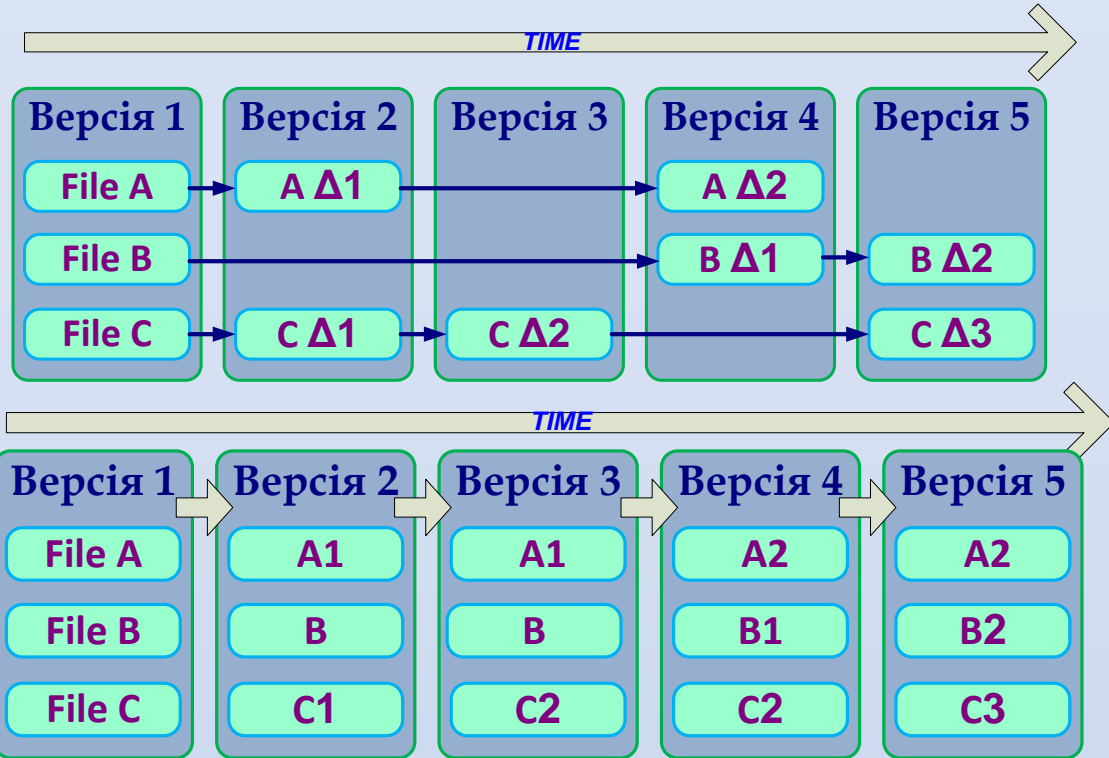
дельта –контроль
версій файлів.

Git –

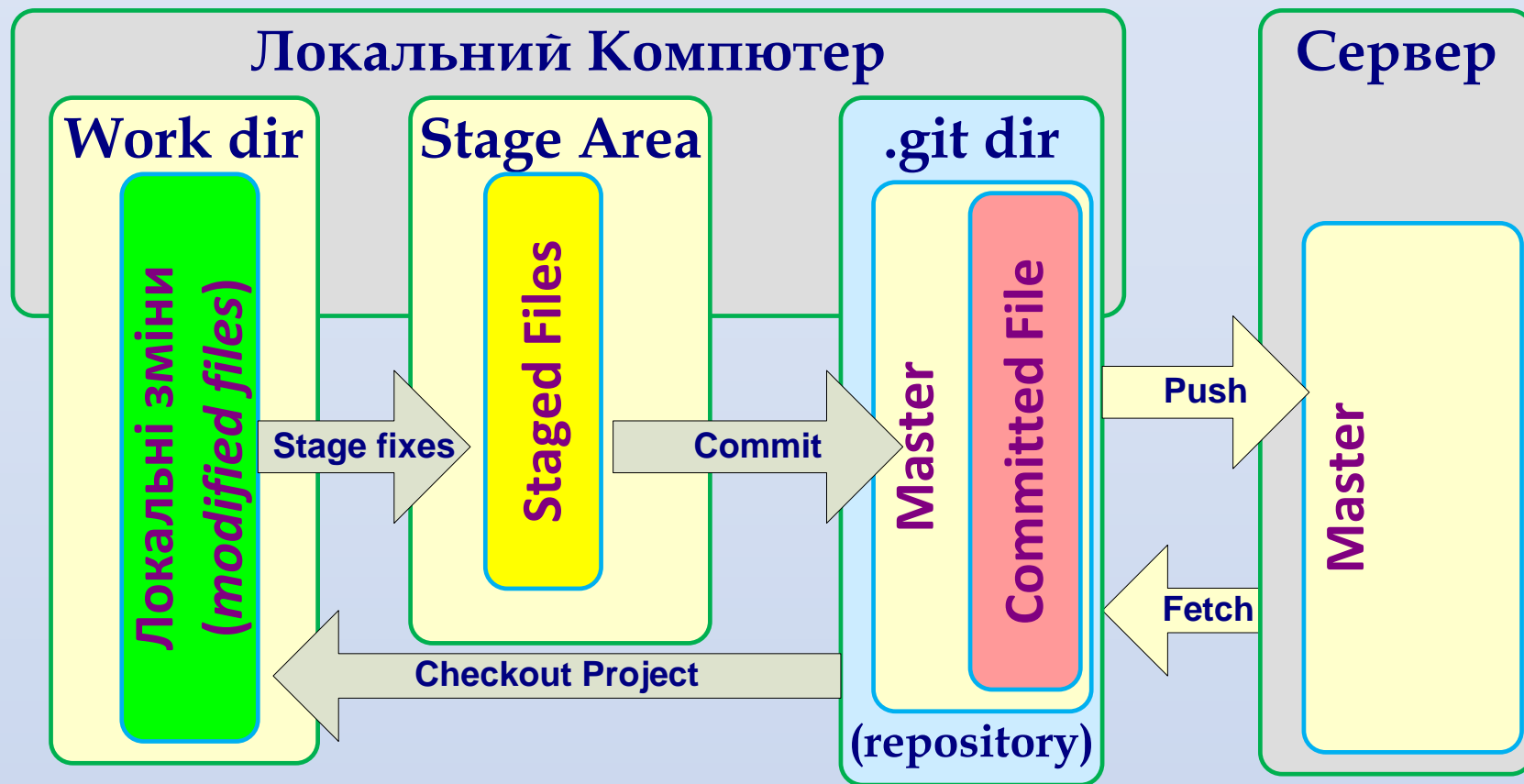
Зберігання

«знімків» (зліпків,
snapshots) версій
файлів.

- Нема необхідності зв'язку з іншими комп'ютерами, повна історія на **кожному** локальному комп'ютері. Локальна робота.
- Перед збереженням, формується контрольна сума (хеш, SHA-1, 40 символів), за якою можна посилатися на «знімок».
- Неможливо змінити файл чи директорію так, щоб це було невідомо системі.

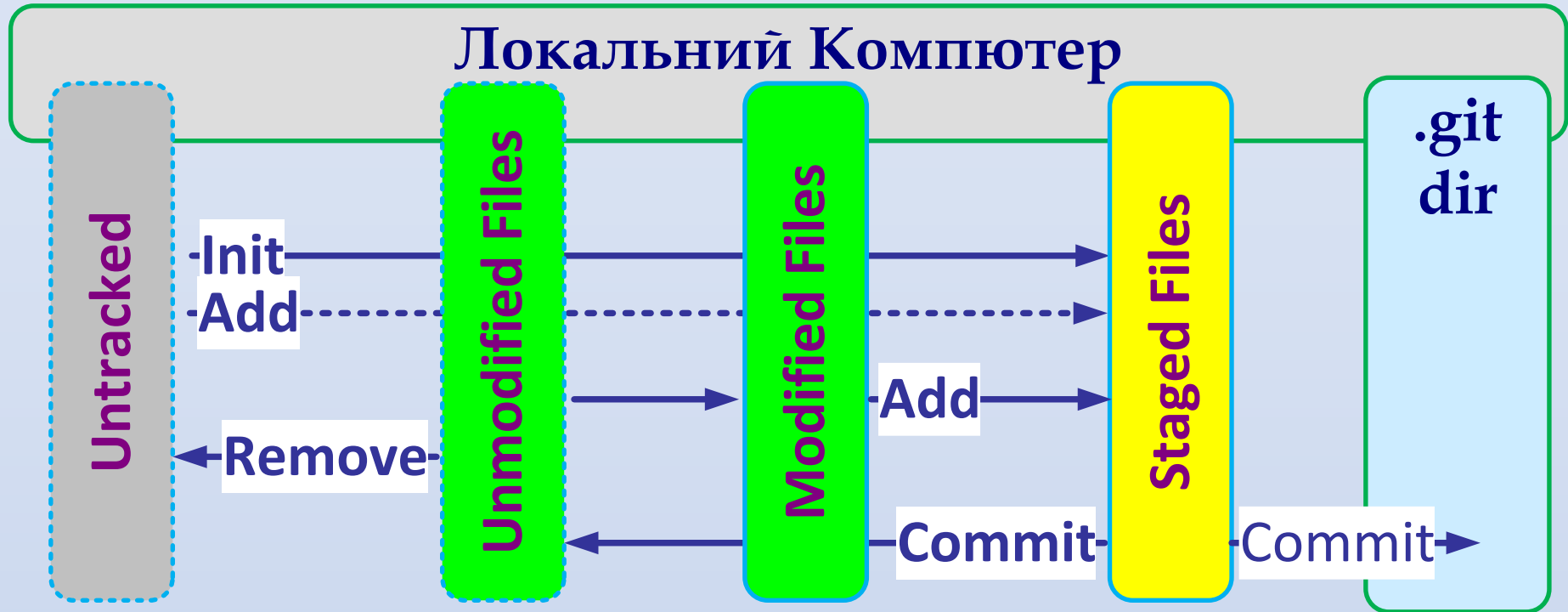


Git. Стани файлів



- Змінений (modified) – до файлу внесені зміни.
- Індексований (staged) – змінений файл, що **буде** внесений до наступного знімку.
- Збережений (committed) – внесений до знімку. 9

Git. Контроль файлів



- *init* – створення локального (пустого) git репозиторію.
- *add* – додавання файлу до stage (індексація).
- *commit* – фіксує «знімок» в репозиторії.
- *log* – історія коммітів.

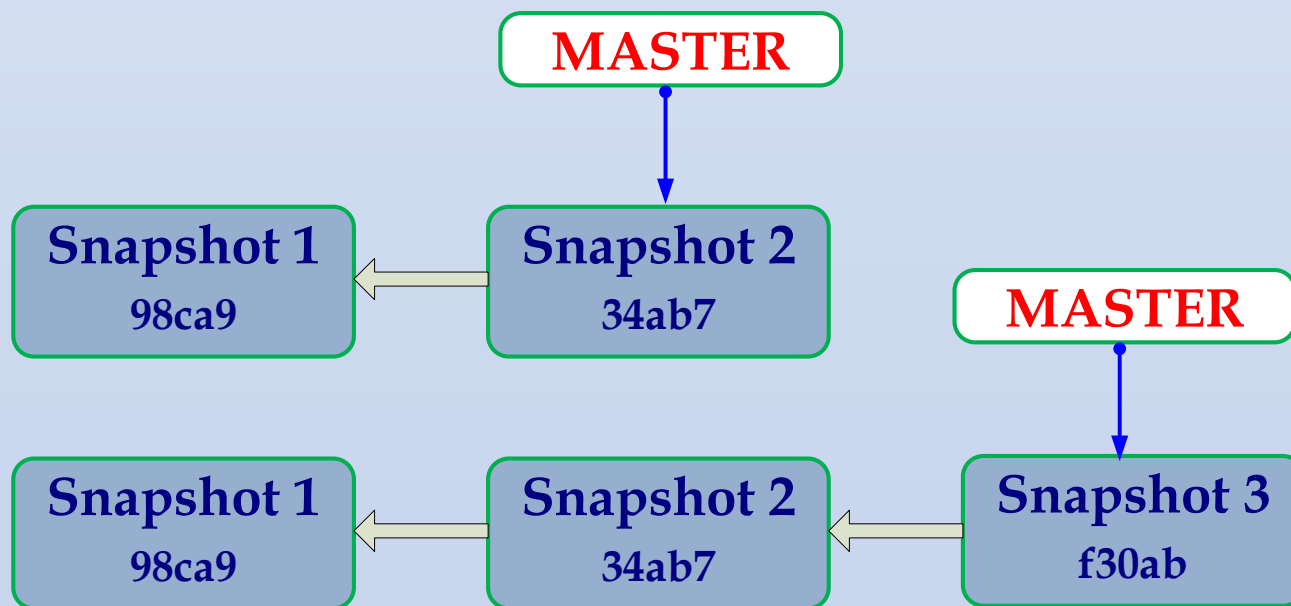
Git. Зберігання знімків



COMMIT → Git обчислює хеш кожної робочої директорії (файлу) та зберігає ці об'єкти дерева в сховищі Git. Потім Git створює об'єкт `snapshot`, що зберігає метадані та вказівник на корінь дерева проекту. Це дозволяє відтворити цей знімок, коли потрібно.

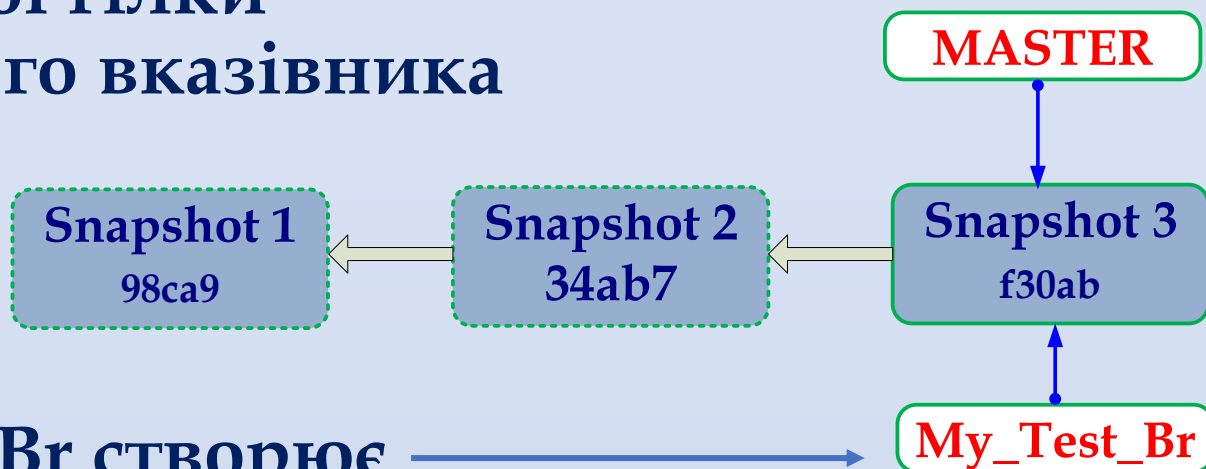
Git. Гілки (branches)

Гілка - вказівник, що може пересуватись та посилатись на одну з фіксацій (snapshots).
За замовчуванням ім'ям першої гілки в Git є **master**. Коли виконується commit, вона переміщується вперед автоматично.



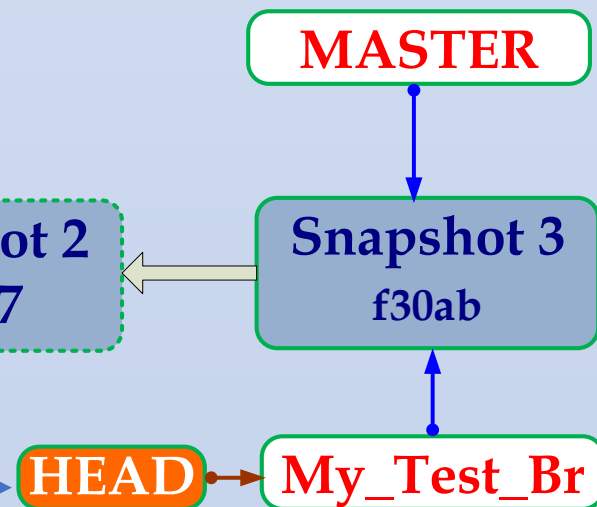
Git. Гілки (branches)

Створення нової гілки –
створення нового вказівника



Спеціальний вказівник HEAD
вказує на поточну гілку.

checkout My_Test_Br пересуває
HEAD

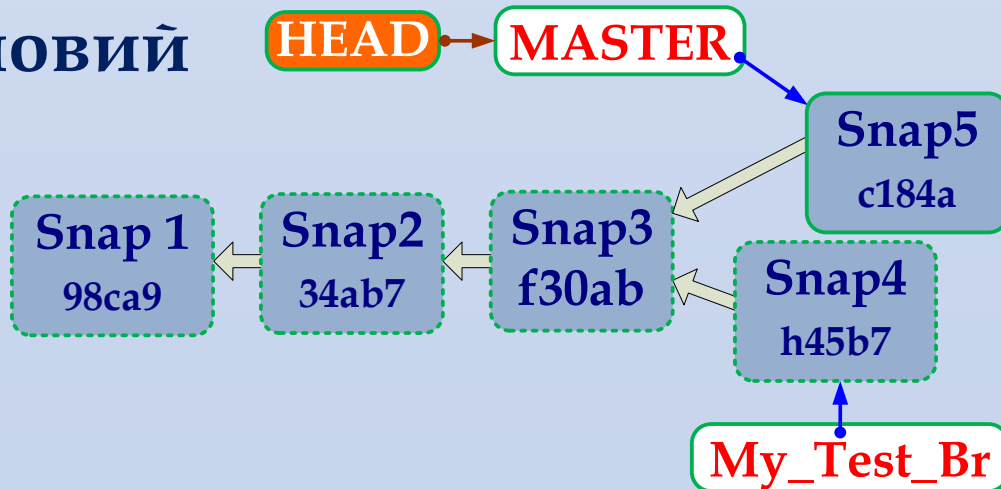
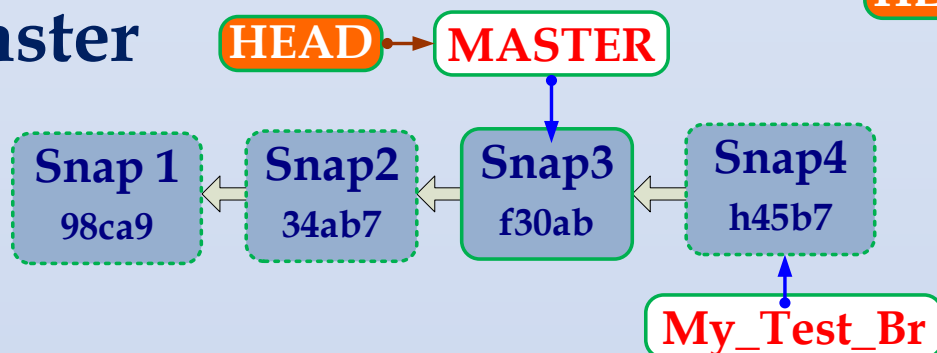
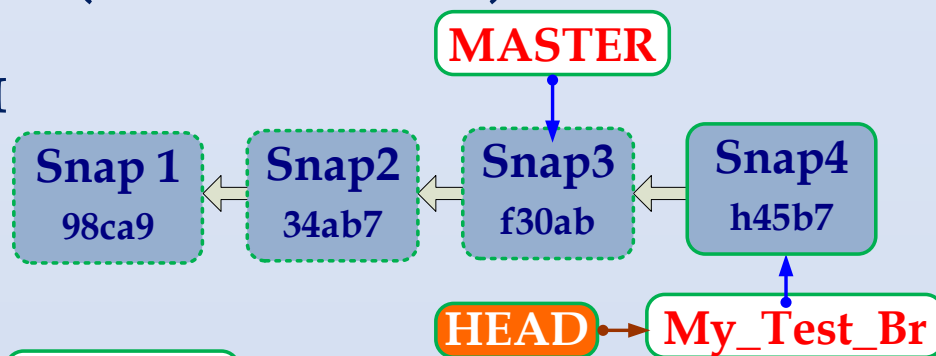


Git. Гілки (branches)

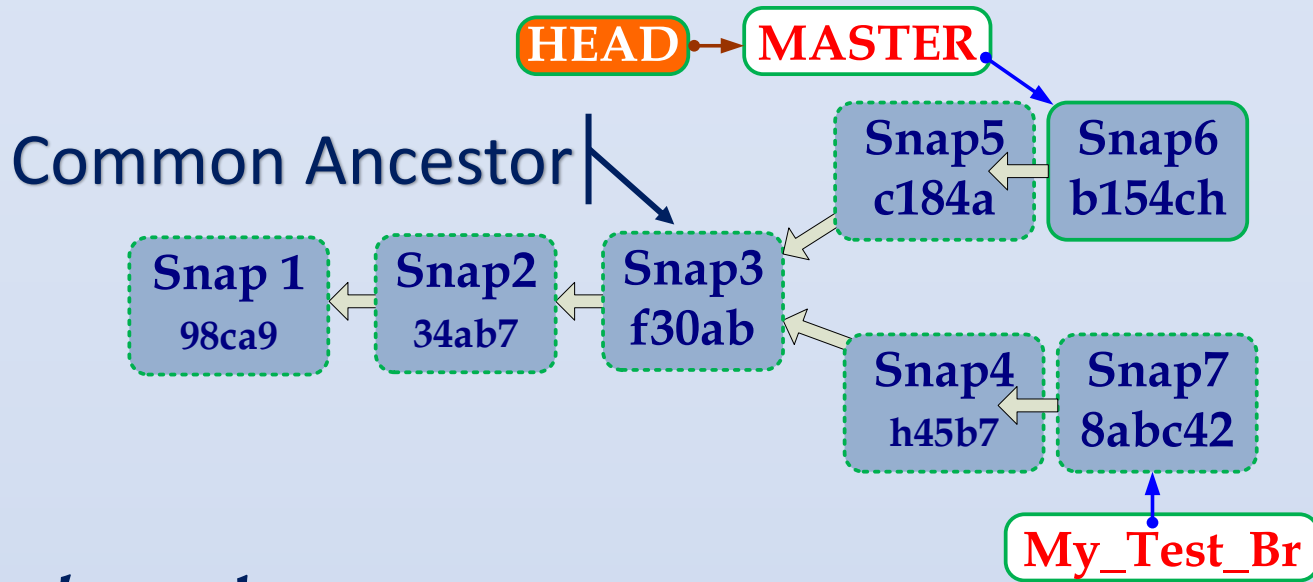
Новий commit - новий snapshot (4)

checkout Master

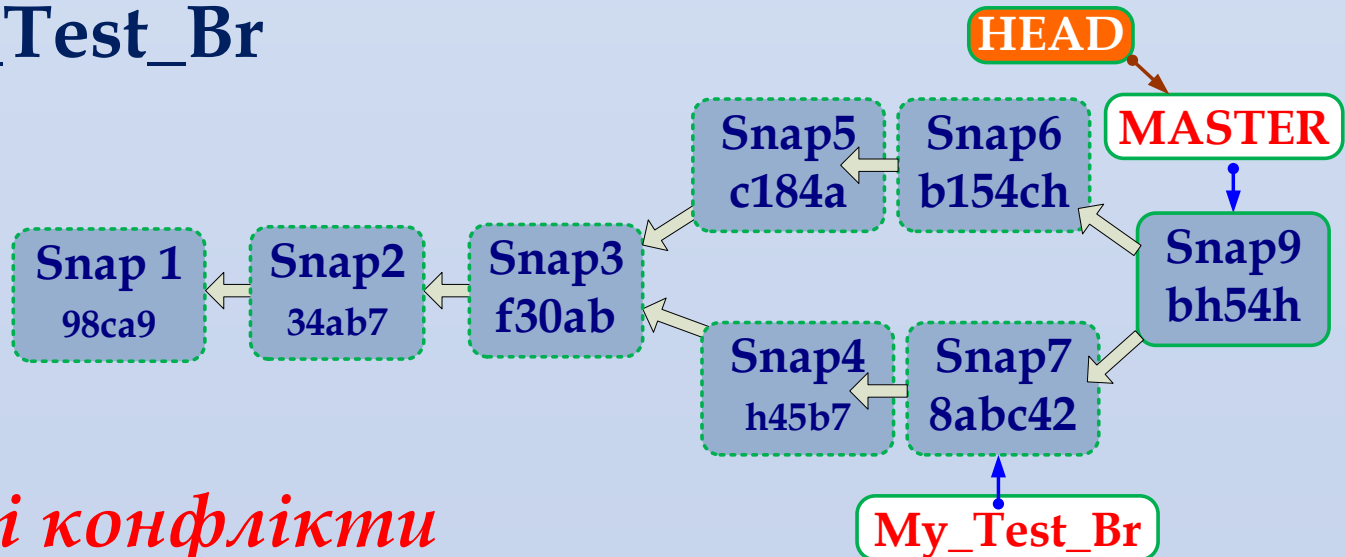
Новий commit - новий snapshot (5)



Git. Гілки, злиття

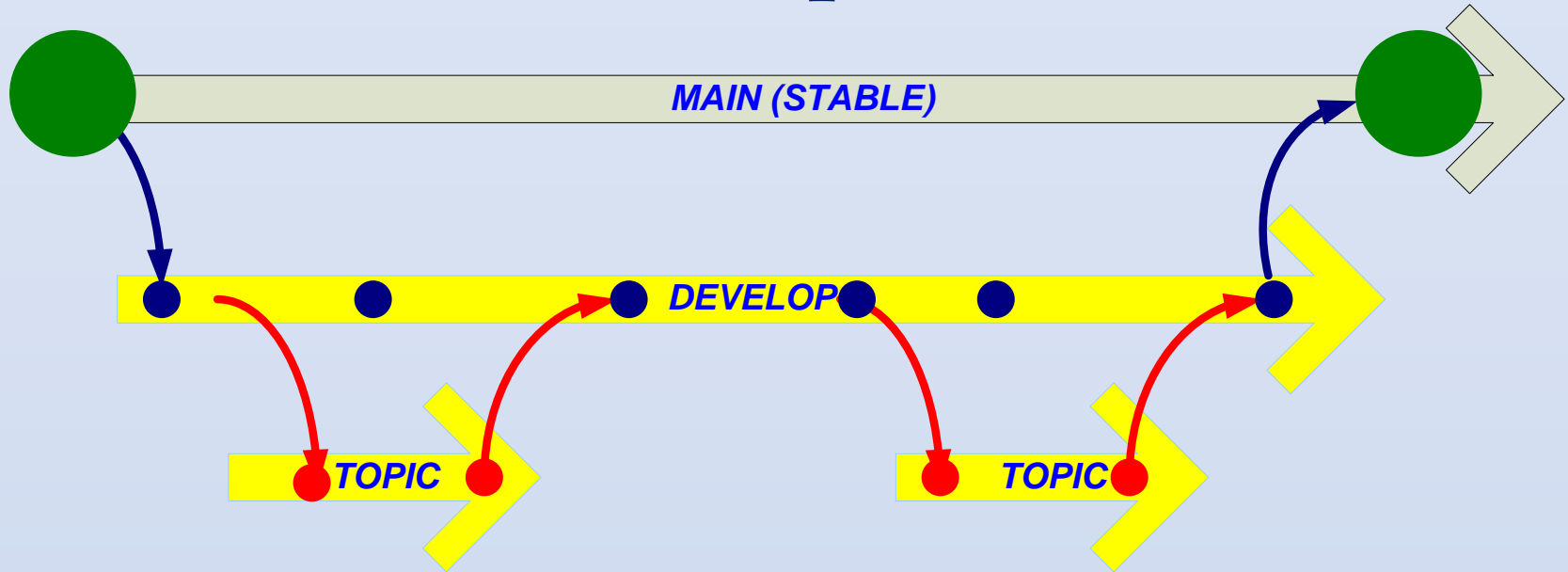


checkout master
merge My_Test_Br



!! Можливі конфлікти

Git. Гілки, тривалість

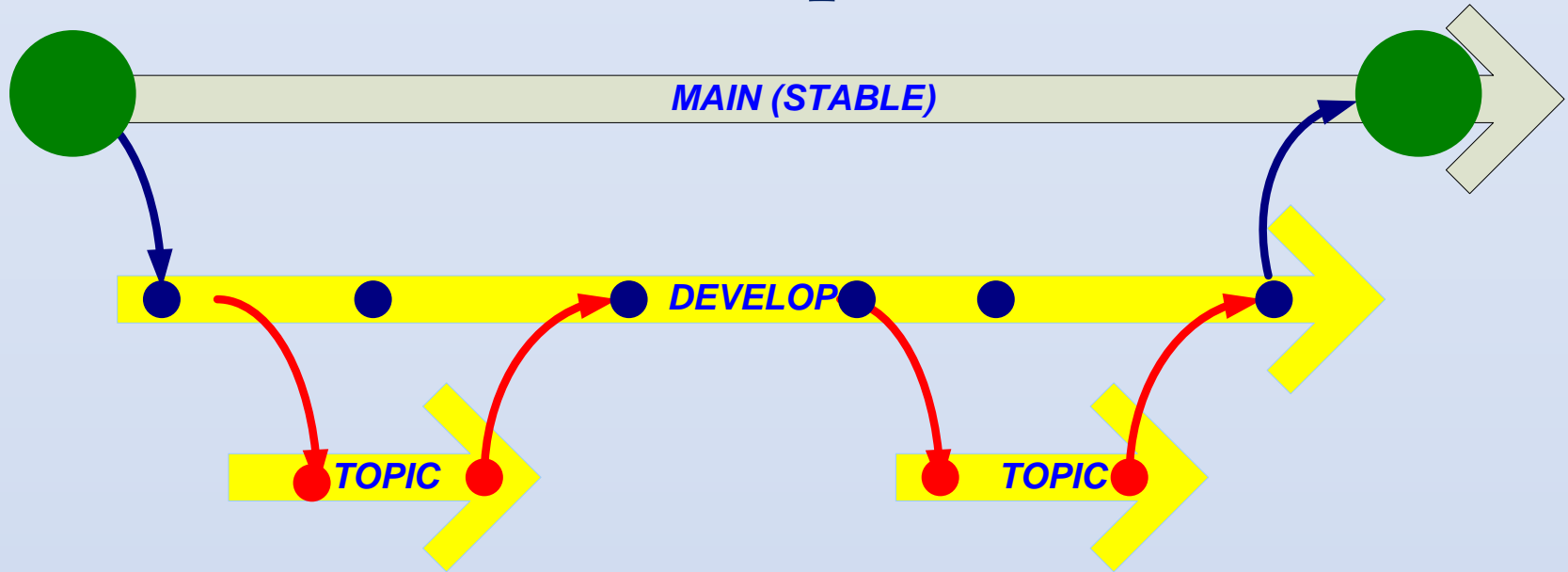


Гілка **MASTER** - стабільна версія.

Гілка **DEVELOP** - гілка тестування стабільності.

Гілка **TOPIC** - розробник, виправлення .

Git. Гілки, тривалість



Гілка **MASTER** - стабільна версія.

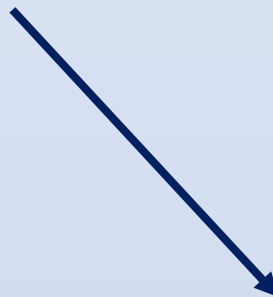
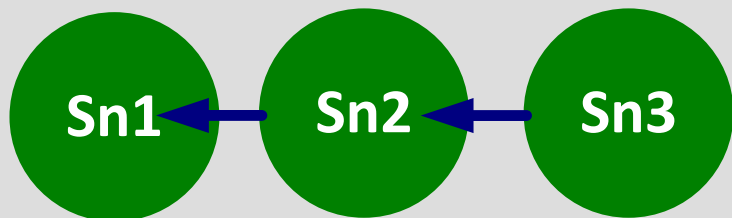
Гілка **DEVELOP** - гілка тестування стабільності.

Гілка **TOPIC** - розробник, виправлення .

Git. Віддалено-відслідковувані гілки

<відділене_сховище>/<гілка>

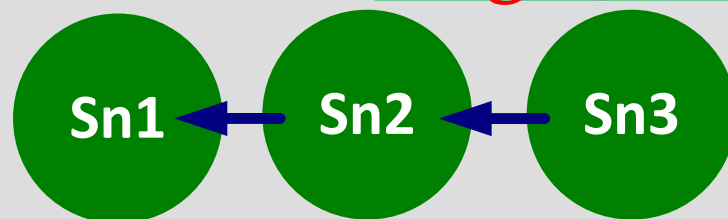
Server **origin** **master**



Клонування до
локального комп'ютеру
clone <шлях до
репозиторію>

Local

origin/master

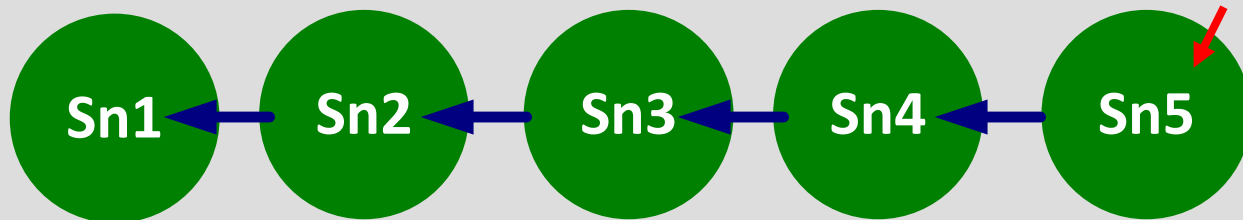


master

Git. Віддалено-відслідковувані гілки

Server **origin**

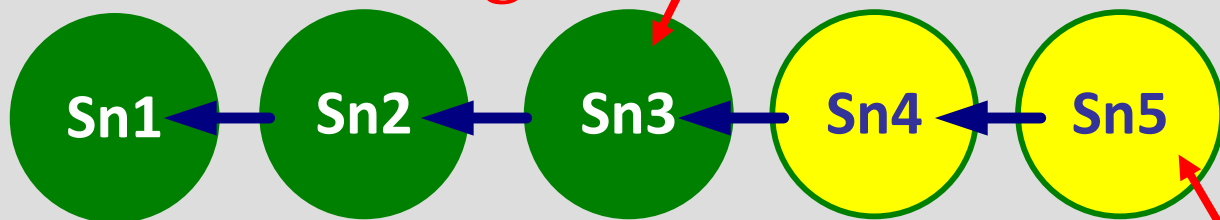
master



Різний прогрес – різні історії відділеної гілки та локальної гілки

Local

origin/master

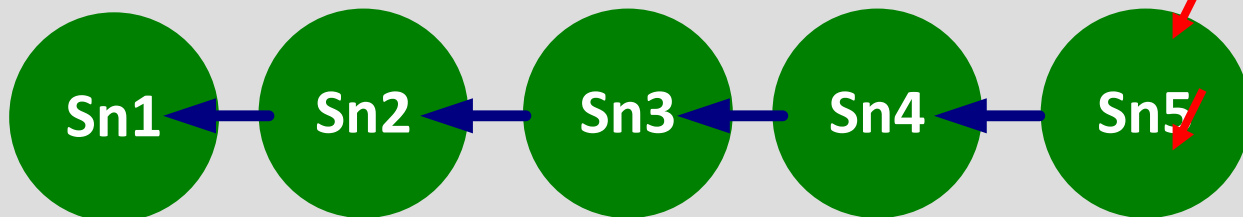


master

Git. Віддалено-відслідковувані гілки

Server **origin**

master

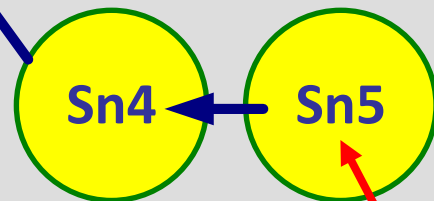
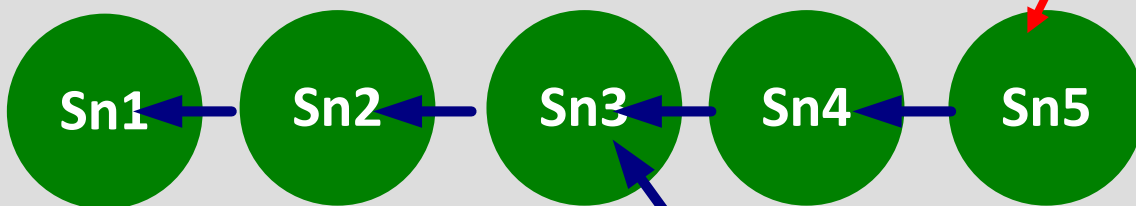


Синхронізація- команда *fetch* - оновлює віддалене посилання

На локальний **master** не впливає. Зливання - *merge*

Local

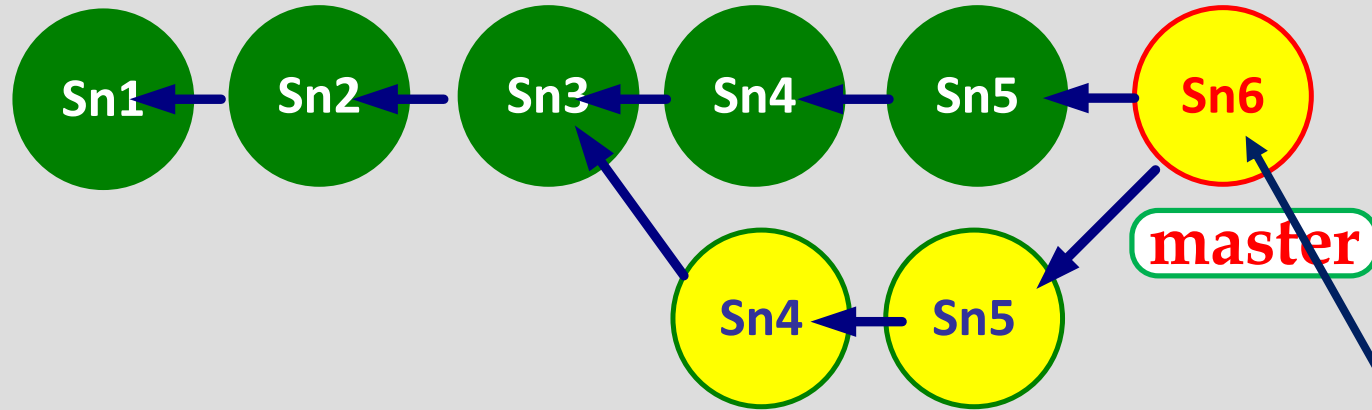
origin/master



master

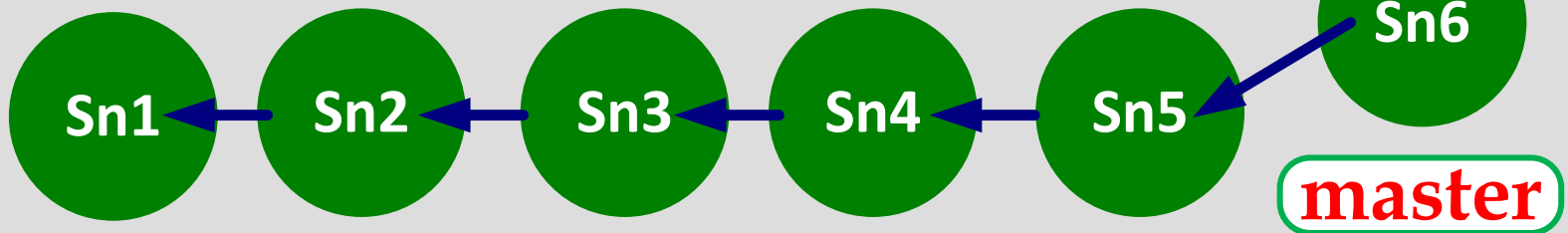
Git. Надсилання

Local



Надсилання - команда *push* - оновлює віддалене посилання (коли є право !!!)

Server **origin**



Git. Інструментарій

Командний рядок

Git SCM <https://gitforwindows.org/>

Графічний інтерфейс для Win & macOS

GitHub Desktop - <https://desktop.github.com/>

Вбудований Git клієнт в Visual Studio

Git хостинг

GitHUB - <https://github.com>

Сервері рішення Git

WebGitNet - <https://github.com/otac0n/WebGitNet>

GitHUB. Корисні посилання

Free Computer Science University

<https://github.com/iteachmachines/Free-Computer-Science-University>

CS Books

<https://github.com/AB1908/CS-Books>

A curated list about Python in Education

<https://github.com/quobit/awesome-python-in-education#roadmaps++>

Source Code for the Book Classic Computer Science Problems in Python

<https://github.com/davecom/ClassicComputerScienceProblemsInPython>

Рекомендована ЛІТЕРАТУРА

- **Томашевський О.М., Цегелік Г.Г., Вітер М.Б., Дудук В.Ш. Інформаційні технології та моделювання бізнес-процесів. Навч. посіб. – К.: «Видавництво «Центр учбової літератури», 2012. – 296 с.**
- **Карпенко М.Ю., Манакова Н.О., Гавриленко І.О. Технології створення програмних продуктів та інформаційних систем. Навч. посіб. - Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім.О.М. Бекетова, 2017. – 93 с.**
- **Алексенко О.В. Технології програмування та створення програмних продуктів. Конспект лекцій. – Суми, Сумський державний університет, 2013. – 133с.**

Рекомендована ЛІТЕРАТУРА

- **Чакон С., Штрауб Б. Git для професіонального программіста.** – СПб.: Питер, 2016. – 496 с.
- Є українська версія <https://git-scm.com/book/uk/v2>

Посилання

- **GitHub Help** <https://help.github.com/en>
- **Git Magic (Магія Git) // Ben Lynn.**
<http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/uk/ch10.html>

Контрольні запитання

- Надайте призначення систем керування версіями та поясніть відмінності локальних, централізованих та децентралізованих СКВ. Визначте переваги та недоліки кожного типу.
- Поясніть принцип зберігання версій в СКВ GIT.
- Визначте стани та процеси контролю файлів в СКВ GIT.
- Поясніть процеси створення та злиття гілок версій в СКВ GIT.
- Опишіть роботу з гілками версій, що відслідковуються віддалено.

The END
Mod 2. Lec 6.