

Основи роботи з Git

Публікація репозиторію

Git

Після уроку обов'язково



Повторіть цей урок у відео форматі на [ITVDN.com](https://itvdn.com)

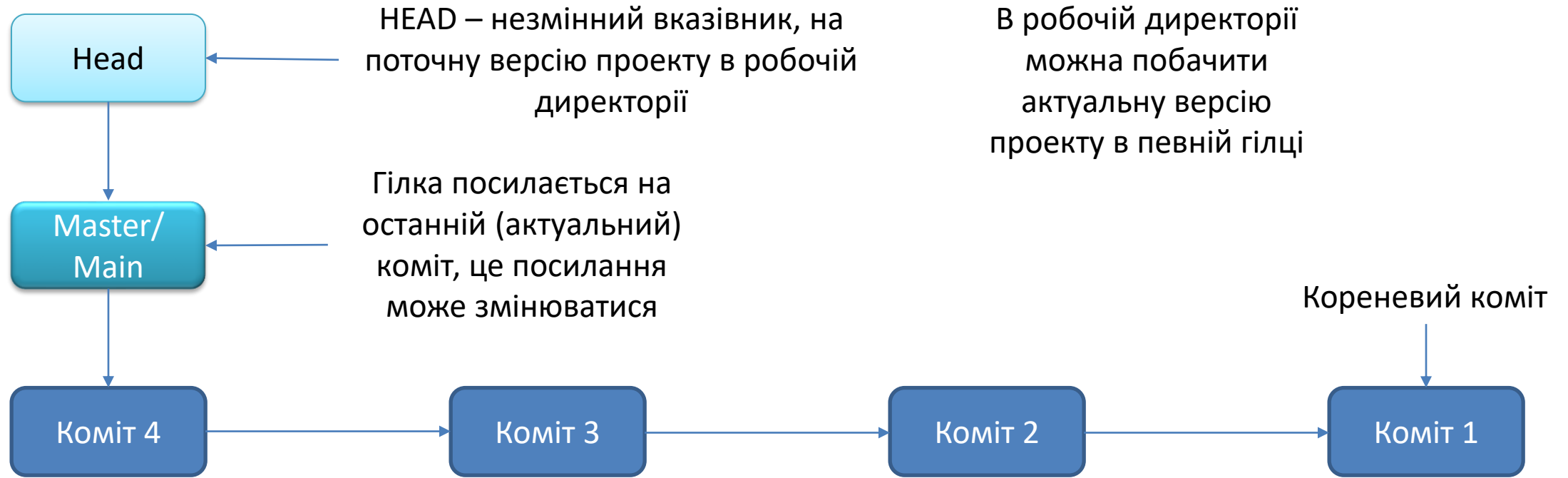
Доступ можна отримати через керівництво вашого навчального центру



Перевірте, як Ви засвоїли цей матеріал на [TestProvider.com](https://testprovider.com)

Основи роботи з Git

Гілка у GIT



Гілка в Git – це простий файл, що містить 40 символів контрольної суми SHA-1 комміта, на який вона вказує; тому операції з гілками є дешевими з погляду споживання ресурсів чи часу. Створення нової гілки в Git відбувається так само швидко і просто як запис 41 байта у файл (40 знаків та переклад рядка).

Основи роботи з Git

Робота з гілками у GIT

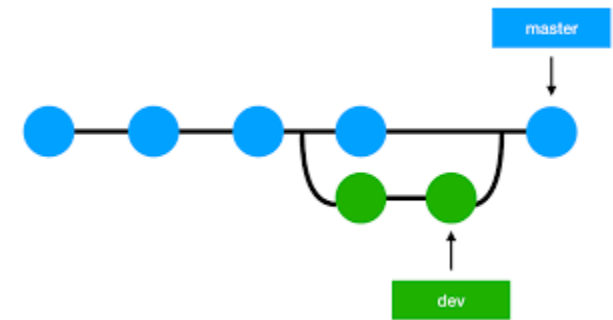
Перехід до певної версії проекту за SHA1-хешем коміту:

git checkout <commit hash> (можна використовувати перші ~5 символів)

Після цього git переміщує вказівник HEAD на певний хеш певного коміту, git після цього візьме певні об'єкти з репозиторію та зробить їх переміщення в робочу директорію.

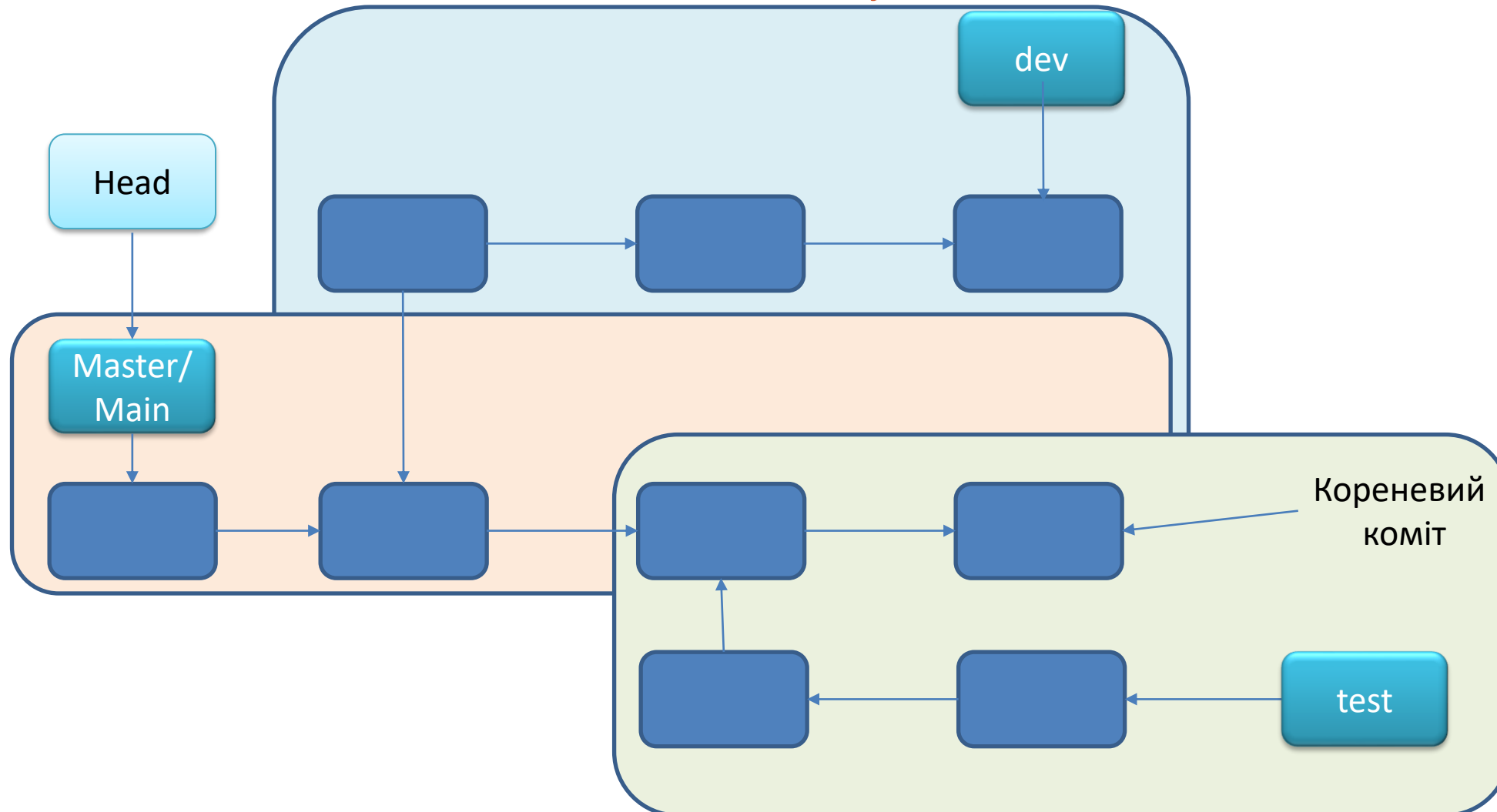
Якщо є декілька гілок, можна виконати перехід в певну версію проекту:

git checkout <branch name>



Основи роботи з Git

Гілки у GIT



Основи роботи з Git

Робота з гілками у GIT

Створення нової гілки (рекомендовано надавати релевантну назву гілці):

git branch <branch name>

+

git checkout <branch name>

=

Створення нової гілки та перейти до неї:

git checkout -b <branch name>

Основи роботи з Git

Робота з гілками у GIT

Отримати перелік всіх гілок:

git branch

Перейменування поточної гілки:

git branch -m <new branch name>

Видалення гілки:

git branch -d <branch name>

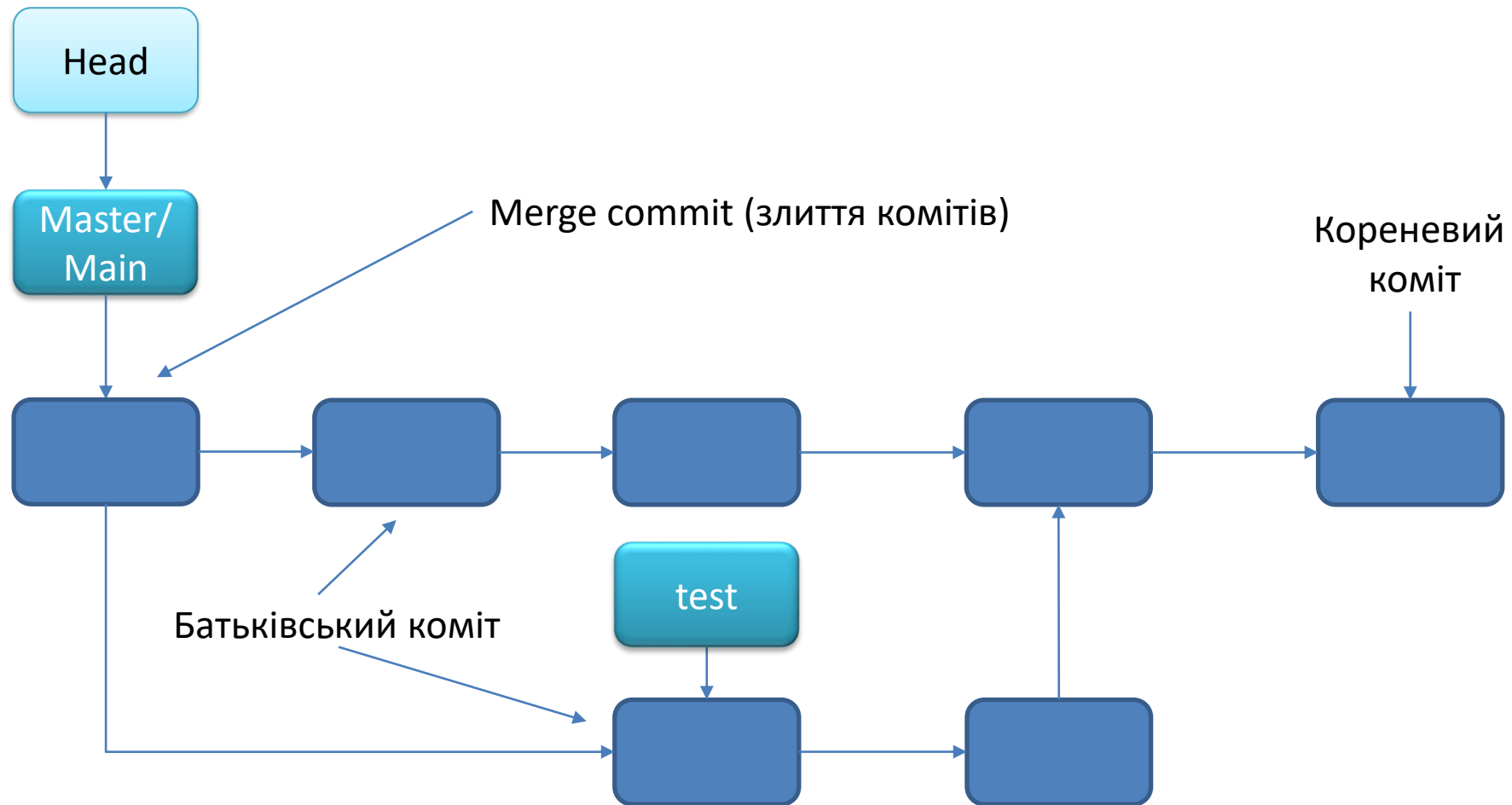
Поточну гілку видалити не можна!

Треба перейти на іншу, а потім видалити необхідну гілку.



Основи роботи з Git

Злиття гілок у GIT



Основи роботи з Git

Злиття гілок у GIT

Злиття іншої гілки(**feature branch**) у поточну(**receiving branch**):

```
git merge <feature branch name>
```

Поточну гілку можна об'єднати з будь-якою існуючою.

Треба перейти на іншу, а потім виконати злиття з необхідною гілкою.

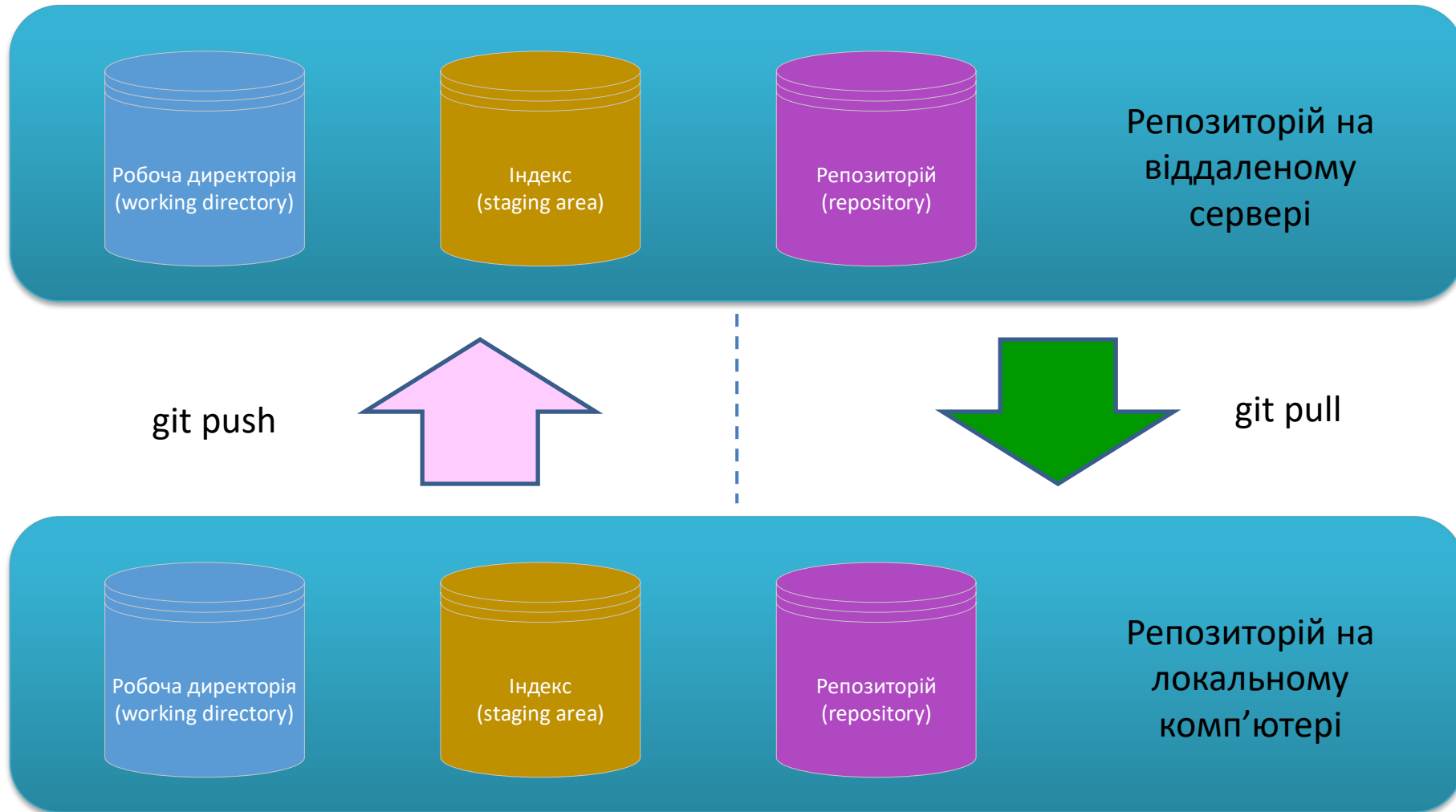
Основи роботи з Git

Алгоритм злиття гілок у GIT

1. Створюємо нову гілку.
2. Виконати перехід у нову гілку.
3. Внести зміни у проект.
4. Створити коміт/декілька комітів у новій гілці.
5. Перейти назад до попередньої гілки, наприклад, main. Також створити коміти.
6. Виконати злиття нової гілки з поточною.
7. Після злиття гілок нову гілку можна видалити.

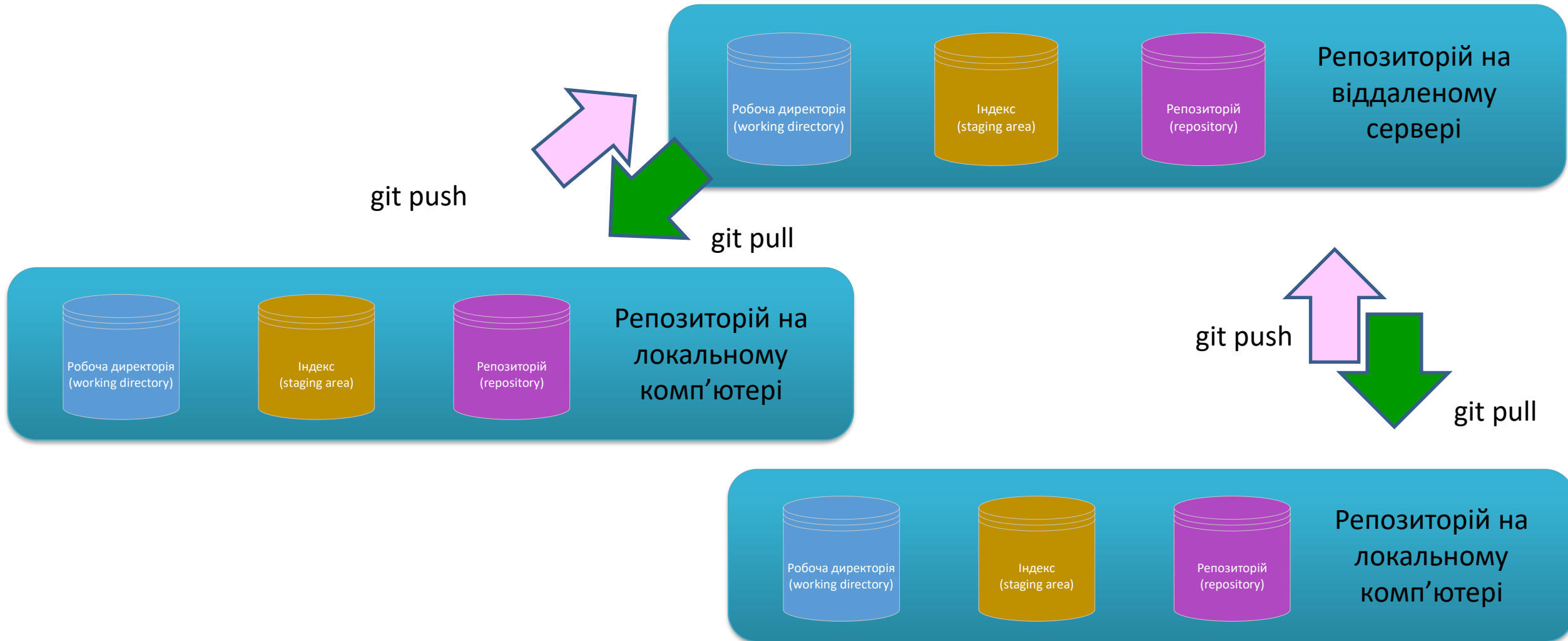
Основи роботи з Git

Робота з репозиторієм на віддаленому сервері GIT



Основи роботи з Git

Робота з репозиторієм на віддаленому сервері GIT



Основи роботи з Git

Робота з гілками у GIT

Відображення усіх гілок, навіть тих, які знаходяться на віддалених репозиторіях:

```
git branch -a
```

Відображення усіх видалених гілок:

```
git branch -r
```

Для перегляду останнього коміту на кожній з гілок, виконайте команду:

```
git branch -v
```

Основи роботи з Git

Робота з гілками у GIT

Опції **--merged** та **--no-merged** можуть відфільтрувати цей список для виведення лише тих гілок, які злиті або ще не злиті в поточну гілку. Для відображення гілок, які ви злили з поточної, можете виконати команду:

```
git branch --merged
```

Для відображення всіх гілок, що містять напрацювання, які ви ще не злили в поточну гілку, виконайте команду:

```
git branch --no-merged
```

Основи роботи з Git

Робота з гілками у GIT

Перейменувати гілку локально можна за допомогою команди:

```
git branch --move bad-branch-name corrected-branch-name
```

Гілка *bad-branch-name* буде перейменована в *corrected-branch-name*, але ця зміна поки що локальна. Для того, щоб усі побачили виправлену гілку у віддаленому репозиторії, треба відправити її туди за допомогою наступної команди:

```
git push --set-upstream origin corrected-branch-name
```

Основи роботи з Git

Робота з гілками у GIT

Зверніть увагу, що поточна гілка ***corrected-branch-name***, яка також є і на віддаленому сервері. Однак, стара гілка все ще там, але її можна видалити за допомогою команди:

```
git push origin --delete bad-branch-name
```

Тепер старе ім'я гілки повністю замінено на виправлене.

Зміна імені гілки, наприклад master/main/mainline/default, зламає інтеграції, служби, допоміжні утиліти та скрипти збирання, які використовувє ваш репозиторій. Перш ніж це зробити, обов'язково проконсультуйтеся з колегами. Також переконайтеся, що ви здійснили ретельний пошук у своєму репозиторії та оновили всі посилання на старе ім'я гілки у вашому коді чи скриптах.



Основи роботи з Git

Робота з гілками у GIT

Перейменуйте локальну гілку **master** на **main** за допомогою наступної команди:

```
git branch --move master main
```

Після цього локальної гілки **master** більше не існує, тому що вона була перейменована на гілку **main**. Щоб решта могли бачити нову гілку **main**, вам потрібно відправити їх у загальний репозиторій. Це робить перейменовану гілку доступною у віддаленому репозиторії:

```
git push --set-upstream origin main
```

Локальна гілка **master** зникла, оскільки вона замінена гілкою **main**. Гілка **main** доступна у віддаленому репозиторії. Стара гілка **master** все ще присутня у віддаленому репозиторії. Інші учасники будуть продовжувати використовувати гілку **master** як основу для своєї роботи, поки ви не здійсніте низку додаткових дій.

Основи роботи з Git

Робота з гілками у GIT

Тепер для завершення переходу на нову гілку перед нами стоять такі завдання:

- Усі проекти, які залежать від поточного, повинні будуть оновити свій код та/або конфігурацію.
- Оновіть конфігурацію всіх тестів, що запускаються.
- Виправте скрипти складання та публікації артефактів.
- Виправте налаштування репозиторію на сервері: встановіть нову гілку за промовчанням, оновіть правила злиття, а також інші налаштування, які залежать від імені гілок.
- Оновіть документацію, виправивши посилання, які вказують на стару гілку.
- Злийте або скасуйте запити на злиття змін, націлені на стару гілку.

Після того, як ви виконали всі ці завдання і впевнені, що гілка **main** працює так само, як гілка **master**, ви можете видалити гілку **master**: `git push origin --delete master`

Основи роботи з Git

Публікація репозиторію

Що таке Gitlab?

Основи роботи з Git

Визначення

GitLab – це система, що дозволяє зберігати свої git-репозиторії на віддалених серверах.

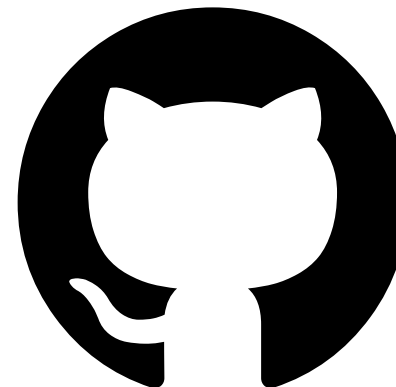


GitLab

Основи роботи з Git

У чому різниця GitLab від GitHub або BitBucket?

Принципово – ні в чому. Кожен із цих сайтів дає можливість зберігати на їхніх серверах свої репозиторії. Є різниця в інтерфейсі та додаткових функціях, які надає сайт (наприклад: CI\CD, створення sub-репозиторіїв та інше).



Основи роботи з Git

Початок роботи

Зареєструйтесь на Gitlab.



GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Sign in	Register
Username or email	
<input type="text"/>	
Password	
<input type="password"/>	
<input type="checkbox"/> Remember me	Forgot your password?
<input type="button" value="Sign in"/>	

Didn't receive a confirmation email? [Request a new one.](#)

[Explore](#) [Help](#) [About GitLab](#)

Основи роботи з Git

Створення ключа SSH

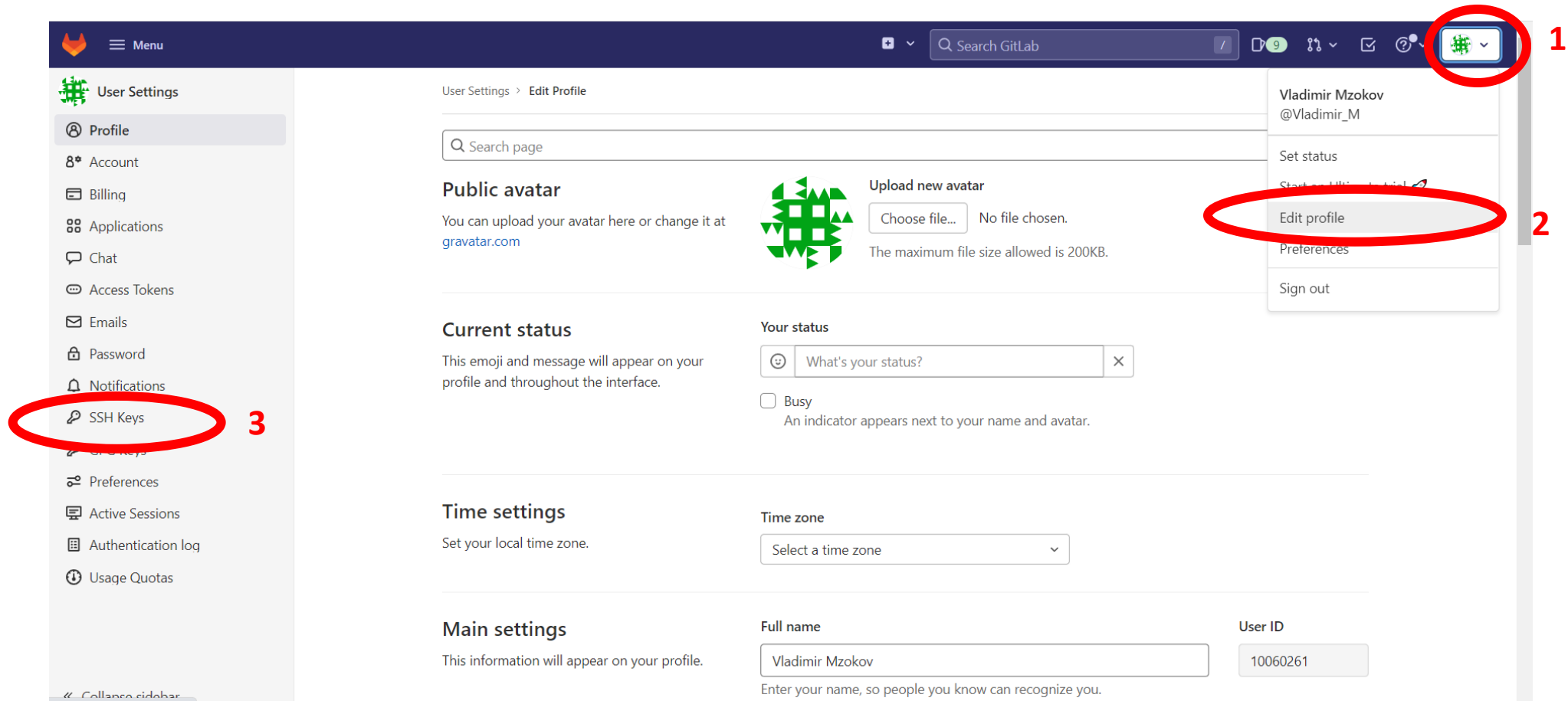
Щоб створити ключ SSH, відкрийте командний рядок та введіть команду, як показано нижче:

```
C:\Users\Vladimir>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Vladimir\.ssh\id_rsa): key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key.
Your public key has been saved in key.pub.
The key fingerprint is:
SHA256:W0MHf3gtoZzRUty+LUacQP3ahIKy0AQQHGddudzf/tM vladimir@DESKTOP-LBIRAI
The key's randomart image is:
+---[RSA 3072]-----+
|  .+++o .ooo=o. |
|  .o  o .+o=+o. |
|    o ..+B++=. |
|  . o.+.oo=.+ |
|  .Soo  + =o |
|  .o .  =.+ |
|    .  . o. |
|              .E |
|              + |
+-----[SHA256]-----+
C:\Users\Vladimir>
```

Основи роботи з Git

Створення ключа SSH

Тепер увійдіть до свого облікового запису GitLab, оберіть опцію «Edit profile», розділ «SSH Keys» :



Основи роботи з Git

Створення ключа SSH

Відкрийте файл `key.pub` та додайте його зміст у поле "Key" та натисніть кнопку "Add key":

User Settings > SSH Keys

Search page

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

SSH Fingerprints

SSH fingerprints verify that the client is connecting to the correct host. Check the [current instance configuration](#).

Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more](#).

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDQCov6YvqMI22ZgsFbNg9Sgms+WriiDTKAGGkMh9vT
4YDUvTiy0yRkl3hrrg4BqfET3TULxOjJaL0BQdhy2HFDJv94CNgbGqu2cUbkQC40Pt/acOS9442LL
25c2vHFcbxXjK6/HvjlypK2CHzoYxxvewLwrQRDwbJSPp3lIsXVPTZu+PORUDDgtULmDupfQcUOD
Gdu9knblCQ0KGzFTWmP4qdQeTEWtaK50dcwEor52p62bKSC/s3KjVq58ryJETDi2wiAbMXwj7nKc
kj4MbOpEmldWlcZeT22bgYUgip1b5d1Nb75Ojh9KUgOERHiCYV3znJI8fhAuoUK7eibD+ptN
MFQRxXksfXt9hP6RiAuGfyiU3v3JvtuEx0FogMSIQo+RUEHNSEUAILbE0HKHBdPOQ7gjmyq2yuXe
```

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

Title

vladimir@DESKTOP-LBIRAIL

Key titles are publicly visible.

Expiration date

2023-08-22

Optional but recommended. If set, key becomes invalid on the specified date.

Add key

Основи роботи з Git

Створення ключа SSH

Після цього на екрані відобразиться наступна інформація:

The screenshot displays the GitLab web interface. The top navigation bar includes the GitLab logo, a menu icon, a search bar, and user profile icons. The left sidebar lists various user settings: Profile, Account, Billing, Applications, Chat, Access Tokens, Emails, Password, Notifications, SSH Keys (highlighted), GPG Keys, Preferences, Active Sessions, Authentication log, and Usage Quotas. The main content area shows the 'SSH Keys' section for the user 'vladimir@DESKTOP-LBIRAI'. It features a search bar and a table with the following details:

SSH Key	
Title: vladimir@DESKTOP-LBIRAI	<pre>ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDCoV6YvqMI22ZgsFbNg9Sgms+WnIiDTKAGGkMh9vT4YDUvvT</pre>
Created on: Aug 22, 2022 7:59pm	
Expires: Aug 22, 2023 12:00am	
Last used on: Never	

Below the table, the 'Fingerprints' section displays the MD5 and SHA256 hashes of the key. A red 'Delete' button is located at the bottom right of the key details.

Основи роботи з Git

Створення нового проекту

Після реєстрації ви побачите сторінку проектів. Створіть новий проект. Для цього натисніть на зелену кнопку New Project.



Основи роботи з Git

Створення нового проекту

Після цього ви побачите форму створення нового проекту.

New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), [among other things](#).

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose **CI/CD for external repo**.

Information about additional Pages templates and how to install them can be found in our [Pages getting started guide](#).

Tip: You can also create a project from the command line. [Show command](#)

Blank project

Create from template

Import project

CI/CD for external repo

Project name

My First Project

Project URL

https://gitlab.com/leonidpodriz

Project slug

my-first-project

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level

☐ Private

Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☒ Public

The project can be accessed without any authentication.

☐ Initialize repository with a README

Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project

Cancel

Основи роботи з Git

Створення нового проекту

Виберіть ім'я для свого проекту та встановіть налаштування приватності "Public" (наша мета поділитися кодом). Після цього підтвердіть створення репозиторію. Ви повинні побачити таке:

The screenshot shows the GitLab interface for a newly created project named 'My First Project'. At the top, a blue notification bar states: 'Project 'My First Project' was successfully created.' Below this, the project name 'My First Project' is displayed with a lock icon, indicating it is private, and the Project ID '20589530'. To the right are buttons for notifications, stars (0), and forks (0). The main section is titled 'The repository for this project is empty' and provides instructions on how to get started. It includes a 'Clone' button and several options to add files: 'New file', 'Add README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', and 'Set up CI/CD'. Below this, 'Command line instructions' are provided for both global Git setup and creating a new repository. The global setup instructions are:

```
git config --global user.name "Leonid Podriz"
git config --global user.email "leonidpodriz@gmail.com"
```

 The 'Create a new repository' instructions are:

```
git clone git@gitlab.com:leonidpodriz/my-first-project.git
cd my-first-project
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```


 Finally, the 'Push an existing folder' instructions are:

```
cd existing_folder
git init
git remote add origin git@gitlab.com:leonidpodriz/my-first-project.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Основи роботи з Git

Надсилання проекту на сервер

Стадію "Git global setup" ми зробили на попередньому занятті. Так як у нас вже є репозиторій, який ми бажаємо опублікувати, використовуємо інструкції з розділу "Push an existing folder". У нашому випадку вони трохи скоротяться до:



```
git remote add origin git@gitlab.com:leonidpodriz/my-first-project.git
git push -u origin master
```

Зверніть увагу, що *git remote add origin git@gitlab.com:leonidpodriz/my-first-project.git* підходить для мого репозиторію. У вашому випадку буде інше посилання.


Основи роботи з Git

Клонування репозиторію з віддаленого серверу на ПК

Клонування локального репозиторію за допомогою http:

```
git clone https://example.com/gitproject.git
```

Адреса
віддаленого
серверу



Щоб клонувати Git-репозиторій SSH, ви можете вказати префікс ssh:// в URL, наприклад:

```
git clone ssh://[user@]server/project.git
```

Або можна використовувати для протоколу SSH короткий синтаксис на кшталт scp:

```
git clone [user@]server:project.git
```

Також ви можете не вказувати ім'я користувача, Git буде використовувати те, під яким ви увійшли до системи.

Основи роботи з Git

Що далі?

Ви можете продовжувати працювати з **git** локально. Як тільки ви готові показати свій код колегам, виконайте команду **git push**, щоб знову відправити зміни на сервер.

Основи роботи з Git

Спільна робота та оновлення проектів

Коли ви готові поділитися своїми напрацюваннями, лише кілька команд допоможуть вам працювати з віддаленими репозиторіями.

Команда **git fetch** пов'язується з віддаленим репозиторієм і забирає з нього всі зміни, яких у вас поки що немає і зберігає їх локально.

Команда **git pull** працює як комбінація команд **git fetch** і **git merge**, тобто Git спочатку забирає зміни із зазначеного віддаленого репозиторію, а потім намагається злити їх із поточною гілкою.

Команда **git push** використовується для встановлення зв'язку з віддаленим репозиторієм, обчислення локальних змін відсутніх у ньому, та власне їх передачі у вищезгаданий репозиторій. Цій команді потрібне право на запис у репозиторій, тому вона використовує автентифікацію.

Основи роботи з Git

Спільна робота та оновлення проектів

Команда **git remote** служить для керування списком віддалених репозиторіїв. Вона дозволяє зберігати довгі URL репозиторіїв у вигляді зрозумілих коротких рядків, наприклад «origin», так що вам не доведеться забивати голову будь-якою нісенітницею і набирати її щоразу для зв'язку з сервером. Ви можете використовувати кілька віддалених репозиторіїв для роботи та git remote допоможе додавати, змінювати та видаляти їх.

Команда **git archive** використовується для упаковки до архіву зазначених коммітів або всього репозиторію.

Команда **git submodule** використовується для управління вкладеними репозиторіями. Наприклад, це можуть бути бібліотеки чи інші ресурси, які використовуються не тільки в цьому проекті. У команди submodule є кілька під-команд "-add, update, sync та ін" для керування такими репозиторіями.

Основи роботи з Git

Публікація репозиторію

Практична частина

Дивіться наші уроки у відео форматі

ITVDN.com



ITVDN

IT VIDEO DEVELOPERS NETWORK

Перегляньте цей урок у відео форматі на освітньому порталі [ITVDN.com](https://itvdn.com) для закріплення пройденого матеріалу.

Курси записані сертифікованими тренерами, які працюють у навчальному центрі CyberBionic Systematics, та іншими висококваліфікованими розробниками.

Перевірка знань

TestProvider.com



TestProvider

TestProvider – це online сервіс перевірки знань з інформаційних технологій. За його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT-спеціаліста.

Після кожного уроку проходите тестування для перевірки знань на [TestProvider.com](https://testprovider.com)

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.

Q&A

Інформаційний відеосервіс для розробників програмного забезпечення

