Лабораторна робота: Bitbucket, Sourcetree

План

1. Інтеграція Bitbucket, Sourcetree

Корисні матеріали:

Як створити локальний репозиторій в Sourcetree: Корисне відео про Sourcetree: https://www.youtube.com/watch?v=nipVUBf0wVw

Робота з Sourcetree: https://bitbucket.org/product/ru/guides/basics/four-starting-steps?tab=1#step-3-basic-branching-with-bitbucket

Створення гілки в Sourcetree: https://www.youtube.com/watch?v=u8M2Bi9i420

видалення комітів та відміна закомічених змін в файлах в Sourcetree: https://www.youtube.com/watch?v=pAvxDMIw8oE

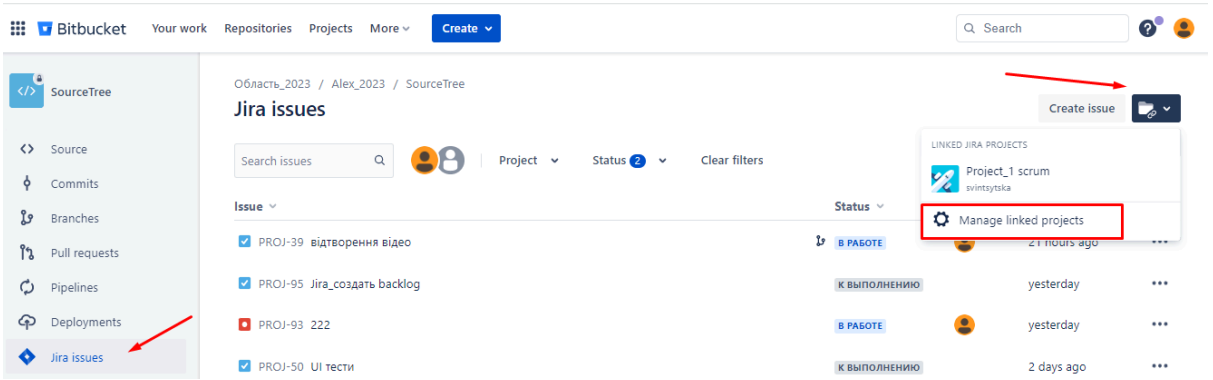
Основи роботи з BitBucket & Jira https://www.youtube.com/watch?v=m-gY5QhmxVI

1. Інтеграція Bitbucket, Sourcetree

1.1. Перейдіть на головну панель інструментів Jira та в лівій частині зверху оберіть продукт Bitbucket. Перейдіть за рекомендаціями для встановлення. Далі в Bitbucket створіть робочу область для управління кодом (це набір проектів).

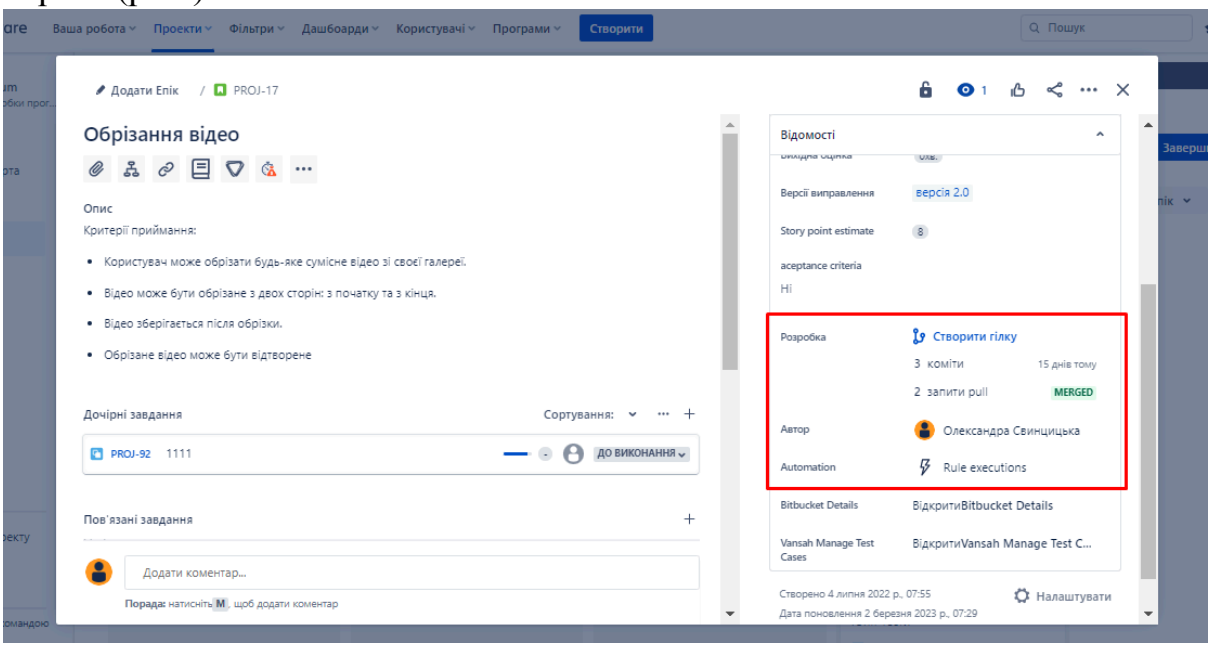
Довідково. *Bitbucket це як GitHab, GitLab або AWS CodeCommit та інші, але для продуктів Atlassian. Bitbucket працює з Git.*

З Bitbucket можна створювати проблеми для Jira, а також їх редагувати. Для цього перейдіть в опцію «створення проблем» (як на рисунку) та оберіть проект для перегляду в Bitbucket. Після налаштувань в розділі «проблеми» Ви будете бачити всі задачі проекту.



В Bitbucket можна створювати commits та pull request, автоматизовувати релізи та ін.. В свою чергу в Jira можна ці процеси

переглядити в самому проекті, можна створювати гілки та commits та pull request (рис.)



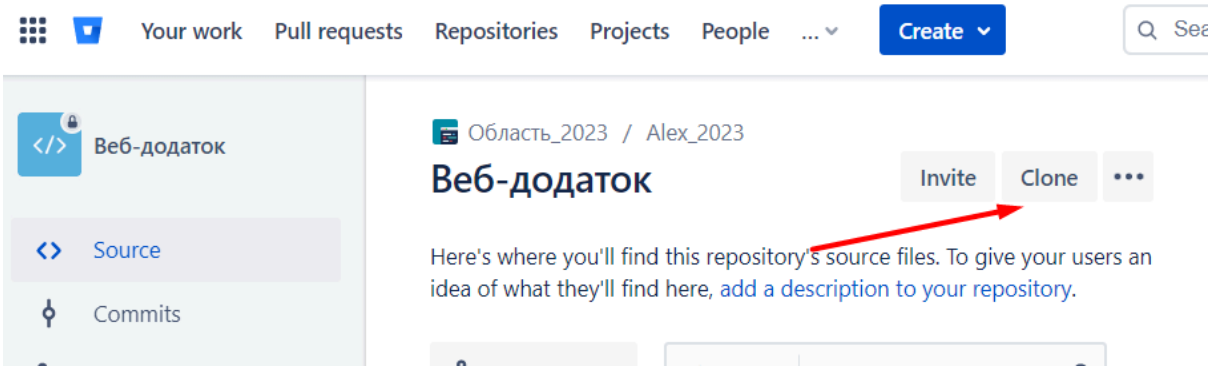
1.2. Для початку роботи створюємо репозитарій в Bitbacket. Натискаємо на опцію “створити репозиторій” на верхній панелі і слідуємо рекомендаціям по створенню віддаленого (основного) репозитарію. Називаємо репозитарій **Веб-додаток (**запамятовуємо шлях папки**)**. За допомогою цього репозиторію буде здійснювати інтеграція з іншими програмами. Розширення файл .git свідчить про відношення до віддаленого репозиторію в Bitbucket. Цей репозитарій як правило є пустим, бо локально з ним не працюють.

Далі ви переходите на сторінку, де відображається вже створений віддалений репозитарій в Bitbacket.

1.3. Наступним кроком є створення локального репозитарію для роботи та його інтеграція з віддаленим. Для інтеграції локального репозитарію з віддаленим, створюємо клон віддаленого в графічній програмі Sourcetree. Йдемо за рекомендаціями.

Використовуємо. Перший шлях клонування в Bitbucket. (рис.)

На панелі інструментів в Bitbucket зверху тиснемо «Clon» репозитарію. Ця дія необхідна для синхронізації. Обираємо обліковий запис, папку (запам’ятовуємо де її створено) і реєструємося.



Далі обираємо Sourcetree (рис.нижче)

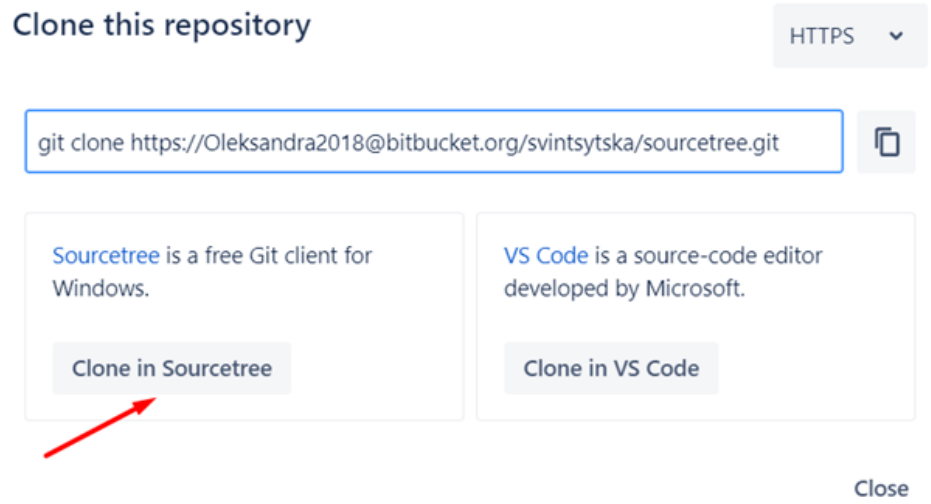


Рис.Створення клону локального репозитарію через Bitbucket для Sourcetree

1.4.Встановлюємо клієнт Sourcetree.

Довідково. Завдяки графічній програмі користувацького інтерфейсу від Sourcetree https://www.sourcetreeapp.com/ працювати з репозитаріями Git стає простіше, а значить, можна зосередитися на створенні коду. Ця програма є клієнтом Git для Windows та Mac, розвиває навики роботи з командним рядком. Отже, можна створювати гілки та нові версії, збирати файли, придатні до використання, здійснювати запит для перегляду перед злиттям в головну гілку.

Встановлюємо Sourcetree, використовуючи той же обліковий запис що і для Jira за рекомендаціями, кроки видно на бічній панелі справа. Обираємо систему Git.

Довідково. Далі буде запит на Ключ SSH (Secure Shell) - це метод автентифікації для з'єднання з віддаленим сервером через мережу, який дозволяє забезпечити безпеку та захист від несанкціонованого доступу.

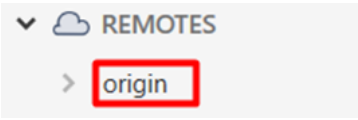
У програмі Sourcetree SSH-ключі використовуються для забезпечення безпеки з'єднання між вашим комп'ютером та репозиторієм на віддаленому сервері. Використання ключа SSH дозволяє вам автоматично аутентифікуватися на віддаленому сервері, без необхідності введення пароля кожного разу при з'єднанні.

Ключі SSH генеруються на вашому комп'ютері і зберігаються у файлі. У Sourcetree ви можете додати свій ключ SSH до налаштувань, щоб мати можливість автентифікуватися на віддаленому сервері з допомогою цього ключа.

Пропонується натиснути «ні»

Після інтеграції в віддаленого репозиторію в Sourcetree, можливо буде запит коду доступу до Git, здійсніть пошук цього коду в налаштуваннях Bitbucket.

В Sourcetree клон має з'явитися у вкладці на бічній панелі.



Може виникнути така проблема, як відображено на рисунку:

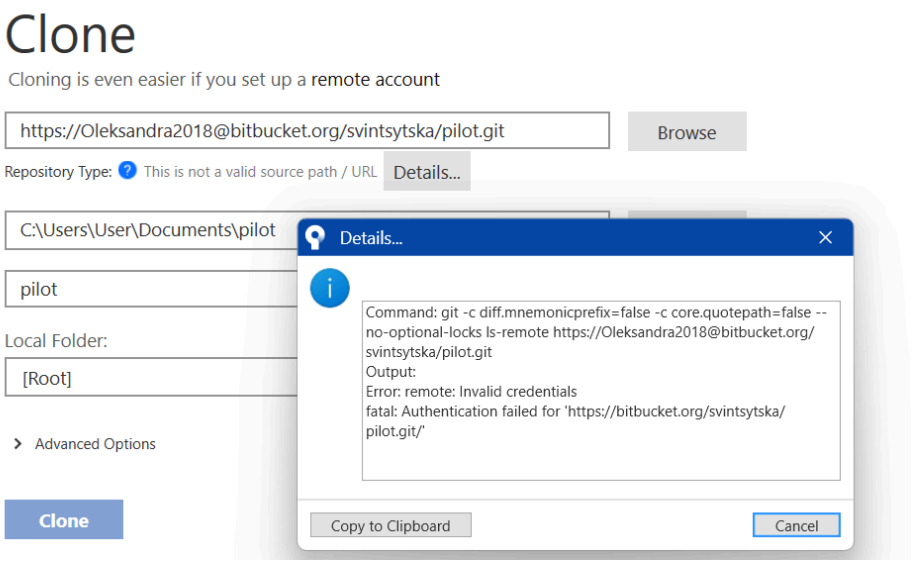
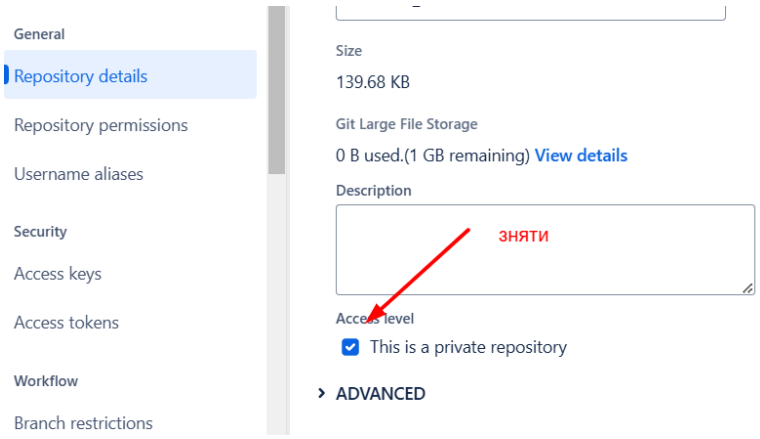


Рис. Це повідомлення свідчить про те, що під час спроби підключення до репозиторію Bitbucket виникла помилка автентифікації.

Можна спробувати змінити налаштування репозиторію (рис. нижче)



Довідково. *Інший шлях клонування в Sourcetree:*

Копіюємо URL віддаленого репозиторію в Bitbucket (див. рисунок вище, отримаємо від натискання елементу clon).

Далі переходимо в Sourcetree і на панелі інструментів обираємо додати клон.

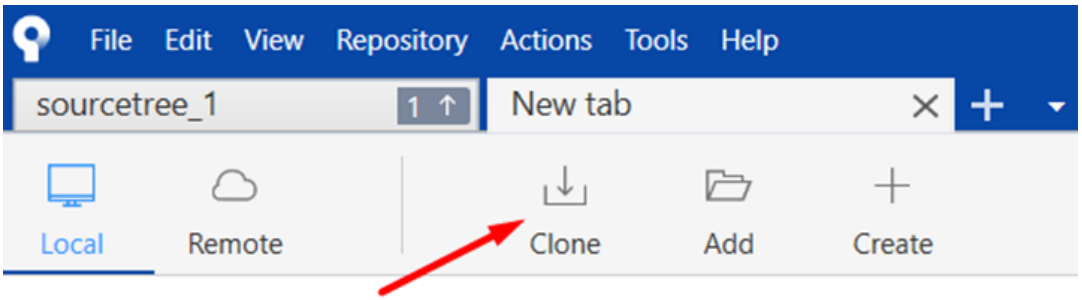
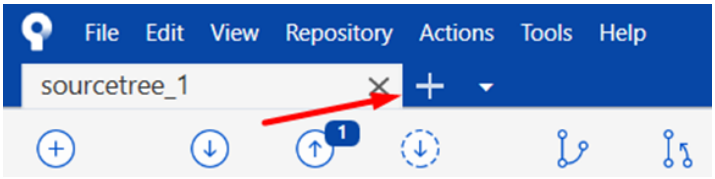


Рис. Додавання клону віддаленого репозиторію в Sourcetree

Вставляємо URL і отримуємо клон віддаленого репозиторцію для синхронізації з локальним репозиторієм. Для цього додаємо між ними зв'язок. Спочатку переходимо до локального репозиторію в Sourcetree, на верхній панелі інструментів обираємо розділ «репозиторій» та функцію додати віддалений репозиторій і натиснути ок. В сучасній версії ця функція автоматизована.

Довідково. Локальний репозиторій можна створити в Sourcetree. Для цього на панелі інструментів тиснемо " + create". Вказуємо шлях до робочого каталогу. Слідуємо рекомендаціям.

Sourcetree підтримує безпосередню інтеграцію з багатьма репозиторіями, сервісами хостингу та системами контролю версій. Наприклад, крім Bitbucket. це GitHub, GitLab та інші.



1.4. На рис. ми бачимо як здійснити синхронізацію двох репозиторіїв.

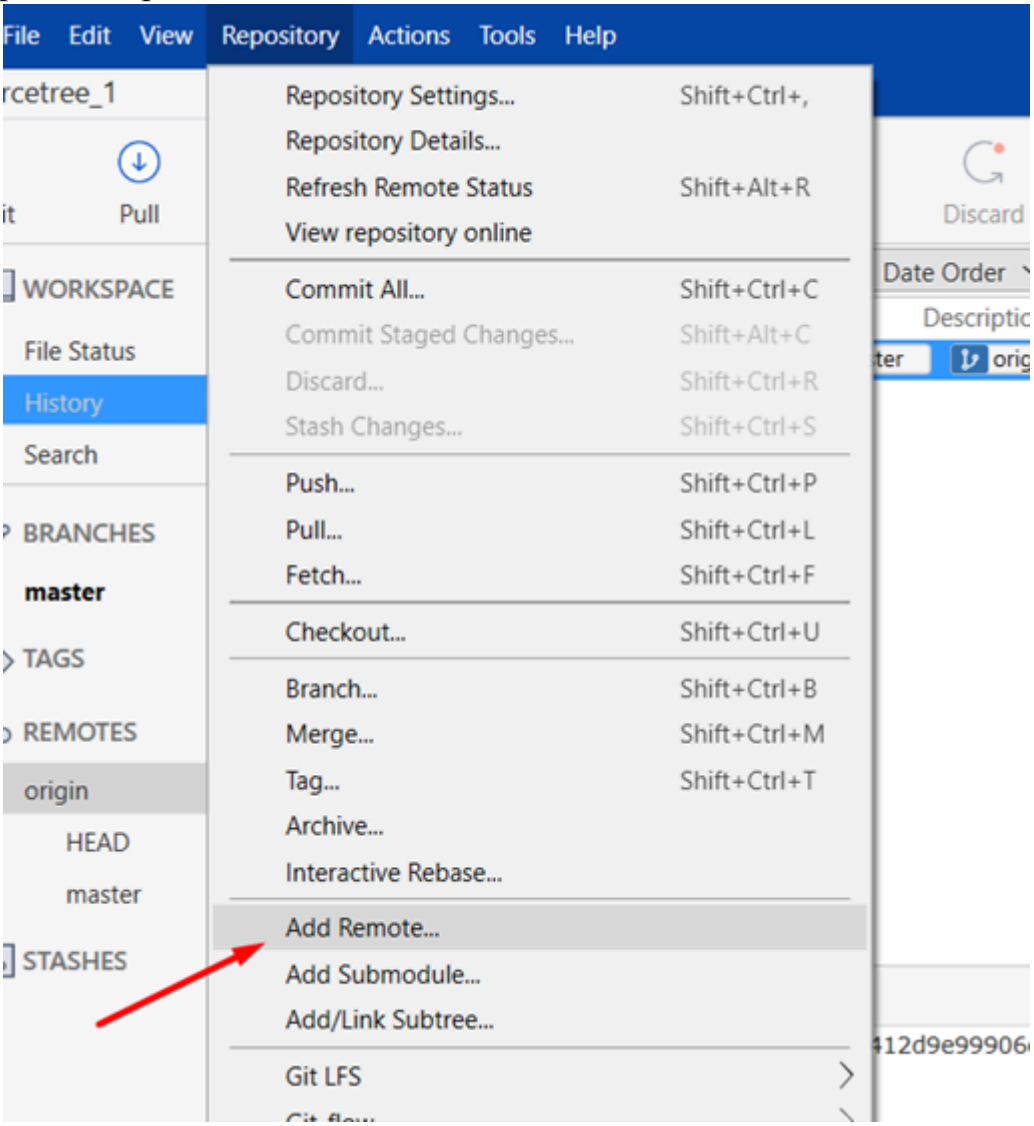


Рис. синхронізація репозиторіїв

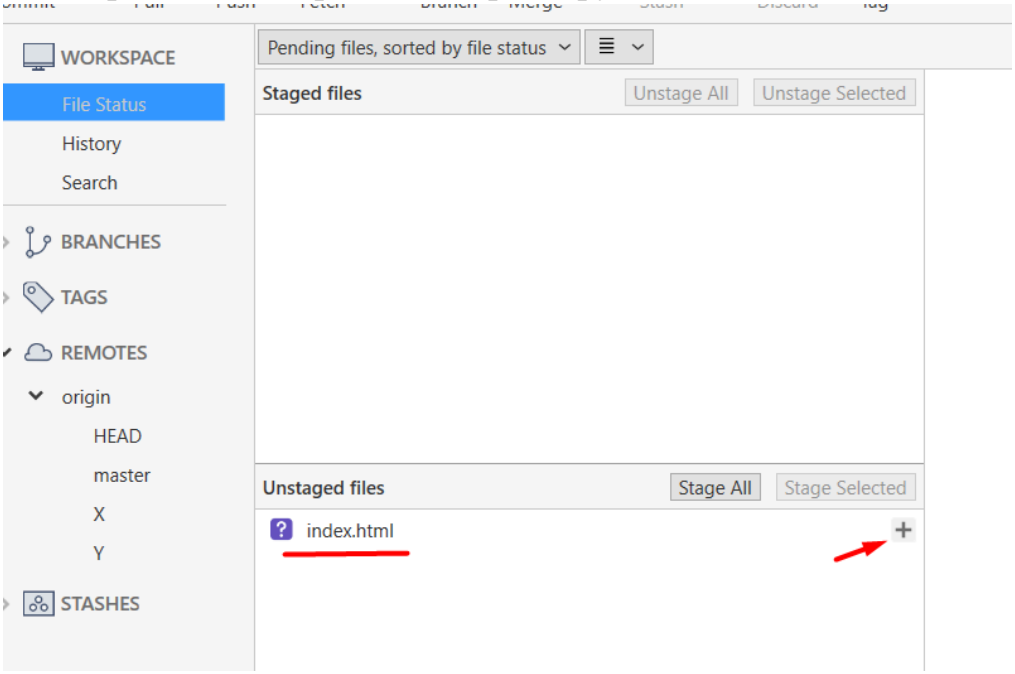
Переходимо в Bitbucket та бачимо синхронізацію.

1.5. Далі перейдіть до робочого каталогу (папку) для збереження даних та (це робоча папка для файлів **Веб-додаток**). Репозиторій буде пустим. Копіюємо туди даний для прикладу файл **index** (проект веб-додатку).

Для отримання досвіду краще пройти цей шлях на власному прикладі. Отже напиши свою програму (наприклад, мікросервіс, моноліт, функція, однорядок). Якщо вам потрібно написати новий код, ви можете відкрити свій проект у своєму обраному текстовому редакторі або середовищі розробки, наприклад, в Visual Studio Code, і зберегти зміни в своєму репозиторії за допомогою SourceTree.

Ви можете працювати у будь-якому середовищі розробки, для цього потрібно здійснити відповідні налаштування в Sourcetree. Ви можете переглядати вміст файлів у своєму репозиторії, робити зміни та коміти змін до репозиторію за допомогою інтерфейсу користувача SourceTree.

1.6. Переходимо до робочого простору *Sourcetree*. Рис. нижче.

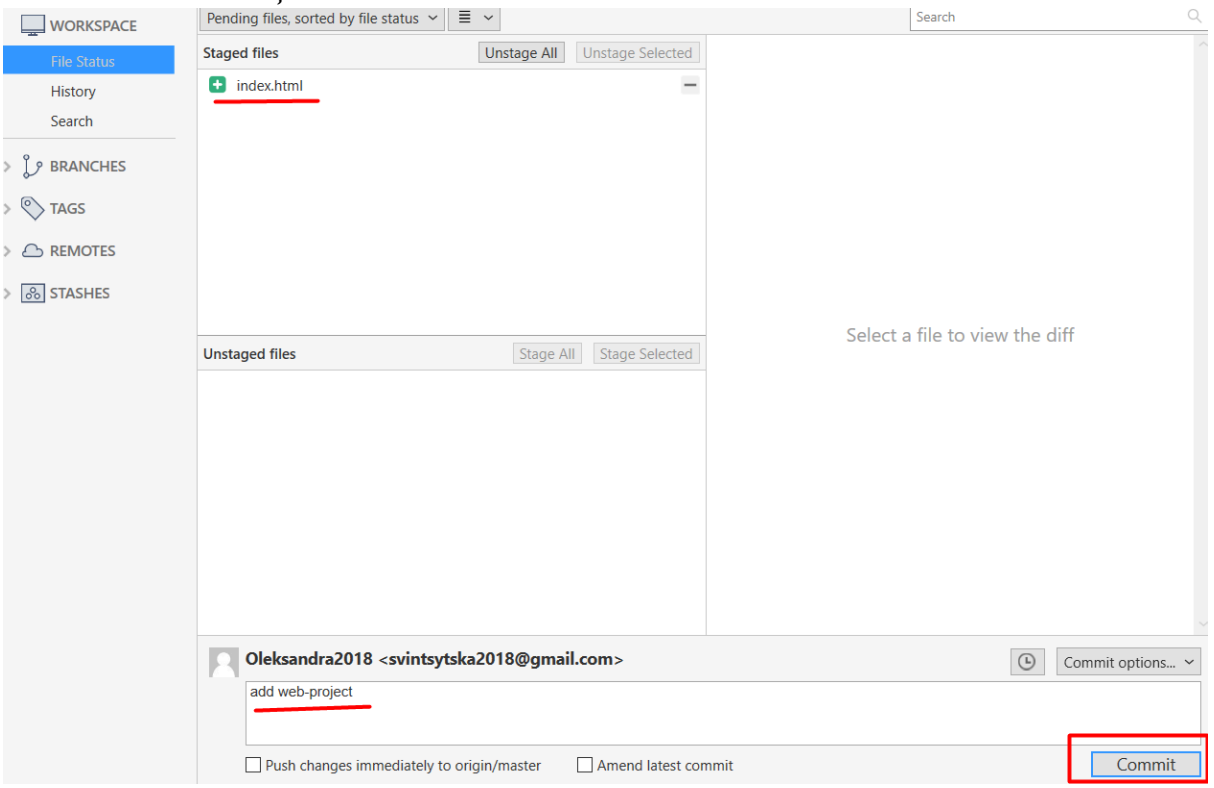


В розділі бічної панелі “file status” відображається стан файлів в робочому дереві (файли проекту) та область підготовки (файли, які увійдуть до наступного commit). Тепер у розділі “file status” Ви буде бачити наш файл проекту **index** (див.рис. вище). Знак питання свідчить про відсутність статусу, оскільки знаходиться в робочому дереві, а не в області підготовки або локальному репозиторії.

Далі готуємо зміст для commit. Тиснемо “+” або справа вверху stage all або … та обираємо stage file (підготовлений файл). Тепер цей файл знаходиться в області підготовки. Таким же чином можна додати усі файли каталогу. В нижньому полі вказуєте також коментар commit (*add web-project)*, яке свідчить про зміни які були внесені в проект або ставите галочку навпроти push (синхронізує цей файл з віддаленим репозиторієм, якщо є незапушені файли, то ви їх будете бачити на панелі інструментів). Це сповіщення стає частиною історії змін. Тепер цей файл передаємо в

локальний репозиторій, як commit, для цього тиснемо «commit» на верхній панелі інструментів для фіксації, щоб у будь-який момент можна було повернутися до цієї версії. Тиснемо *commit.*

Ми ще не створювати гілок, тому всі commit автоматично належать основній гілці master



Довідково. Якщо система видає помилку від push, то можливо відсутня інтеграція з Bitbacket, тоді переходимо в tools на панелі інструментів в розділ options та обрати опцію автентифікація та додати обліковий запис з Bitbacket

Знову переходимо у нашу папку під назвою **веб-додаток** (робоча папка для файлів) і вносимо у наш проект зміни, відкриваємо і змінюємо V7 на V8, додаємо коментар add v.8. Створюємо новий commit.

```
<!-- Header with full-height image -->
<header class="bgimg-1 w3-display-container" id="home">
  <div class="w3-display-left" style="padding:48px">
    <span class="w3-jumbo w3-hide-small">Welcome to DevOps with AWS V7</span><br>
    <span class="w3-xxlarge w3-hide-large w3-hide-medium">Start something that matters</span><br>
    <span class="w3-large">This application is deployed using AWS CodeDeploy</span>
    <p><a href="#about" class="w3-button w3-white w3-padding-large w3-large w3-margin-top w3-opacity w3-ho
  </div>
  <div class="w3-display-bottomleft w3-text-grey w3-large" style="padding:24px 48px">
    <i class="fa fa-facebook-official w3-hover-opacity"></i>
```

Новий commit відображається на головній гілці в Sourcetree

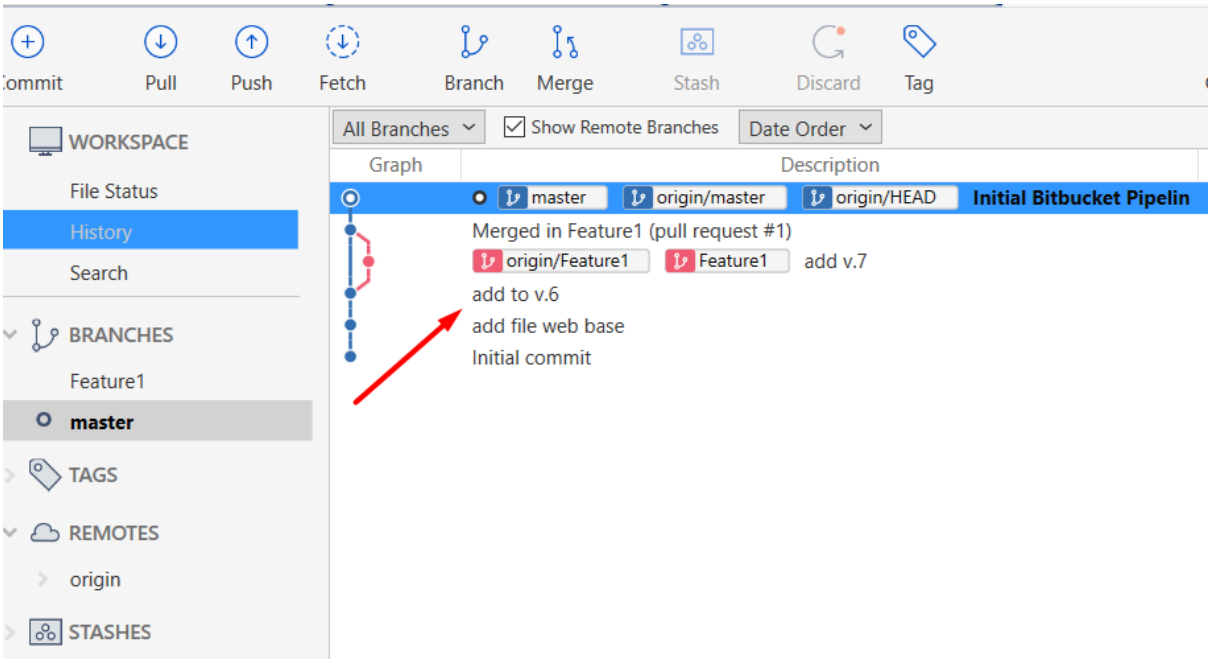


Рис. Огляд робочої області Sourcetree

1.7. Створюємо в Sourcetree тег для commit add V8,, яке і є посиланням на конкретний *commit*. Тег створюється для конкретного *commit* для фіксації його у віддаленому репозиторію. Ми хочемо його закріпити цю версію за міткою commit за допомогою створеного tag. Для цього у Sourcetree клацніть правою кнопкою миші на цьому коміті та виберіть на панелі інструментів TAG, назвіть v8.0 та встановіть прапорець "Push tag" для відображення у віддаленому репозиторії. Ви також можете натиснути тег пізніше як окремий крок. Тепер ви повинні побачити тег v8.0 у коміті у відповідному розділі бокової панелі.

Далі перегляньте *commit* на Bitbucket. В Bitbucket також можна створити commit (див. Рисунок). Зверніть увагу, що з ним пов’язаний тег v8.0.

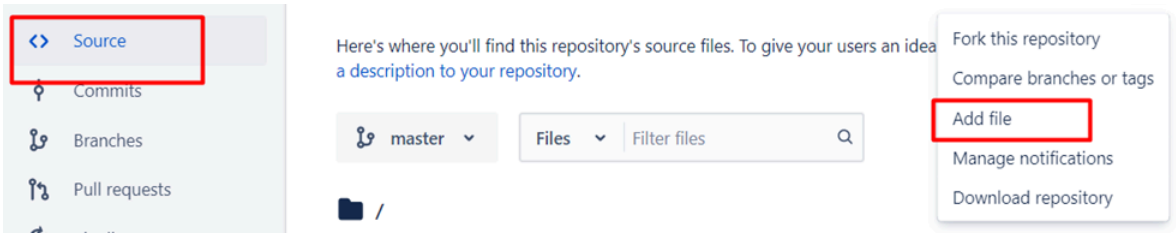
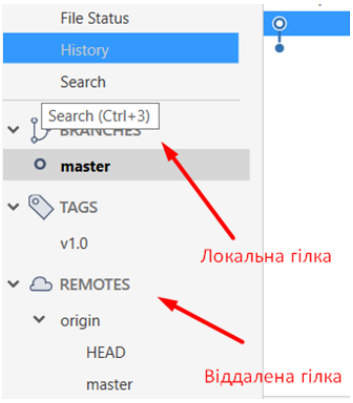


Рис.Створення commit в Bitbucket

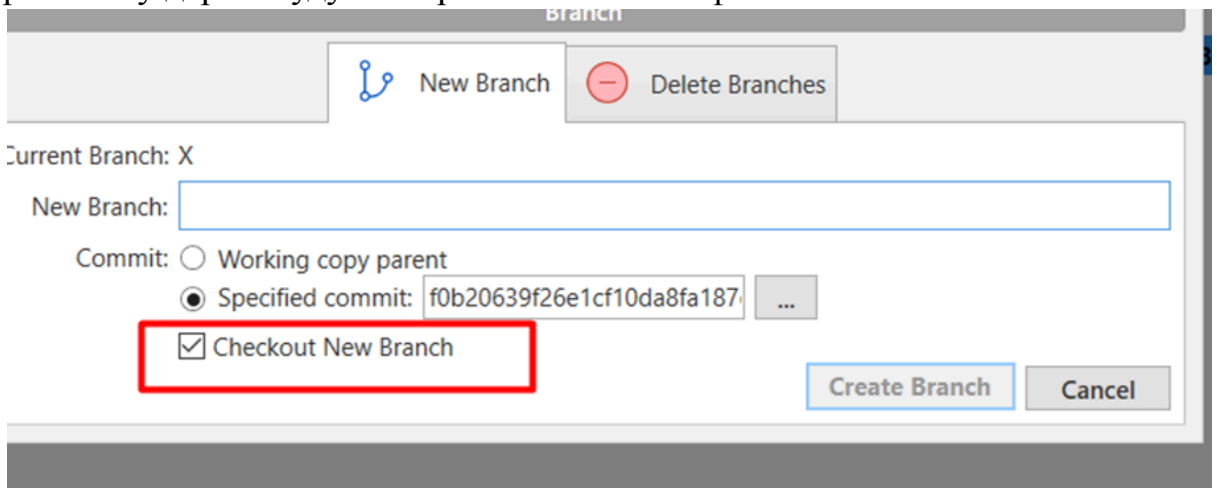
Commit можна редагувати. Більше про цю функцію за посиланням: https://coursera.org/share/38bf6afdc16752eb384f4ee5d544c1d0

1.7. Далі створюємо гілку в Sourcetree. На бічній панелі ми бачимо локальну гілку (master) та віддалену гілку (origin).

Довідково. В Sourcetree, **origin** - це основна (віддалена) гілка, в якій розміщується вихідний код, від якого починаються інші гілки. Посилання **head** вказує на останній коміт в поточній гілці.



Для створення гілки натискаємо на панелі Branch та створюємо гілку під назвою **Feature1** (зазвичай першу гілку називають development, від якої створюються тематичні гілки), тиснемо створити. У випадаючому вікні, *як на рис. нижче,* опція *checkout* (на бічній панелі переходимо на гілку і тиснемо на цю функцію правою клавішею), дозволяє переключатися з гілки на гілки а також синхронізувати дані в робочому дереві. В нашому випадку, це означає переключитись на нову гілку, далі commit буде створюватись у цій гілці. Посилання HEAD буде переміщувати за гілкою з міткою **Feature1**. Вона стає незалежною від основної. А в нашому робочому дереві будуть зберігатися останні файли з основної гілки.



Знову переходимо у нашу папку під назвою **Веб-додаток** (робоча папка для файлів) і створюємо нові зміни у нашому проекті. Відкриваємо і змінюємо V8 на V9, додаємо коментар add v.9 (ми це вже робили вище).

Передивиться зміни, які відбулися в робочому просторі Sourcetree. Якщо переключитися на головну гілку master, то можна знову з нею

працювати. Але гілка master є основною, тому на ній містяться лише основні *commit*, наприклад, для випуску

Далі здійснюємо відправку даних з локального до віддаленого репозиторію. За допомогою функції *push* здійснюється копіювання *commit* з локального репозиторію до віддаленого. При успішному push в них синхронізуються гілки. Всі учасники команди можуть бачити та синхронізувати дані у віддаленому репозиторії.

Отже. переходимо в Sourcetree до функції *push* (на верхній панелі інструментів), перевіряємо дані, щоб всі галочки відповідали потребам та натискаємо push.

Переходимо в Bitbucket в розділ *commits* та бачимо синхронізацію.

У Sourcetree перегляньте історію комітів. Зверніть увагу, що є пов'язані мітки/посилання з комітом(ами).

Отже, наш проект зараз виглядає як на рисунку нижче. В нашому випадку граф лінійний, тому Marge, який ми будемо виконувати в Bitbucket після Pull requests, має пройти безконфліктно і за прискореним методом. Є багато типів конфліктів при злитті, а також обмежені права на злиття.

Перевірте чи всі комміти та гілки відображаються Bitbucket

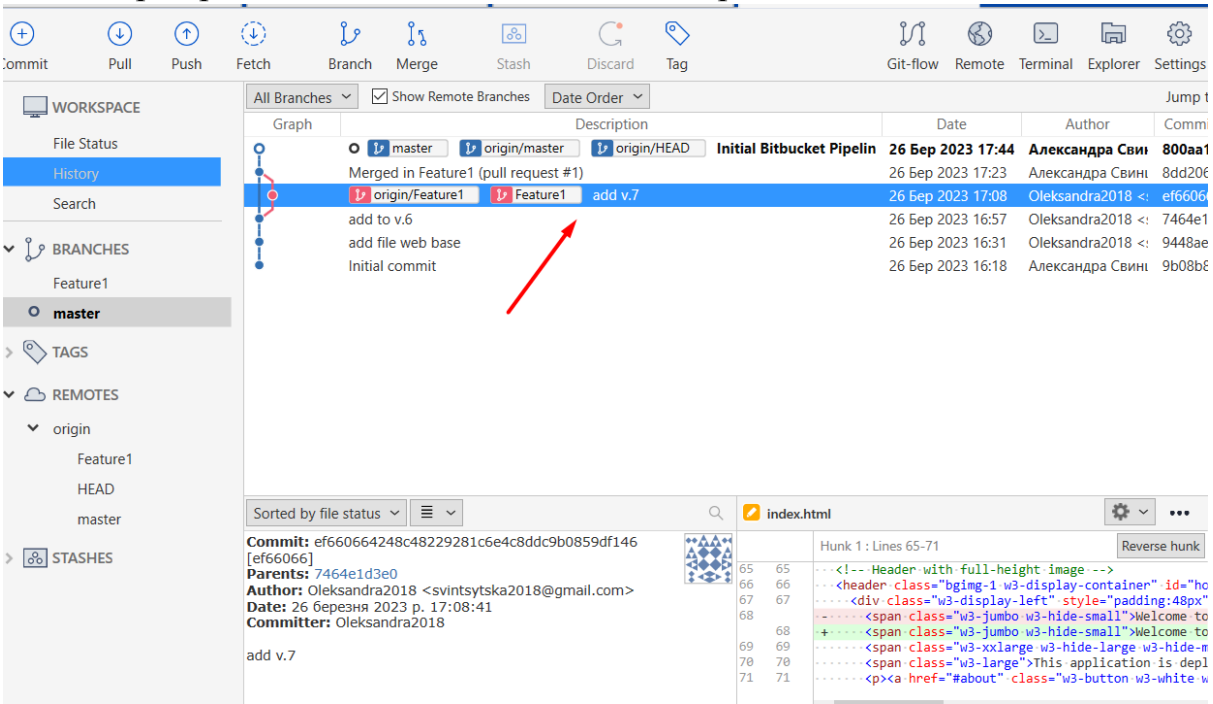
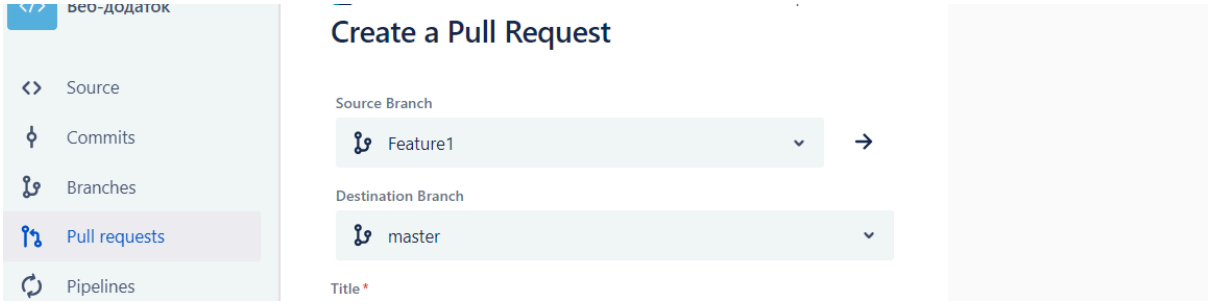


Рис. Створення commit та стан проекту

1.8. Pull requests – запит на злиття гілки в проект. Кінцева мета об'єднати гілки.

Переходимо Bitbucket в розділ Pull requests, тут можна створити або переглянути Pull requests цього репозиторію, тиснемо на “створити” у правому верхньому кутку. З’являється форма для заповнення. Ось таким чином виглядає:



У формі поставити галочку, видалення старої гілки після Pull requests

Заходимо в Pull requests та обираємо опцію Approve (затвердити). Тиснемо. Тепер ви у складі тих, хто погодив цей Pull requests. Тут можна внести зміни, натиснувши опцію edit.

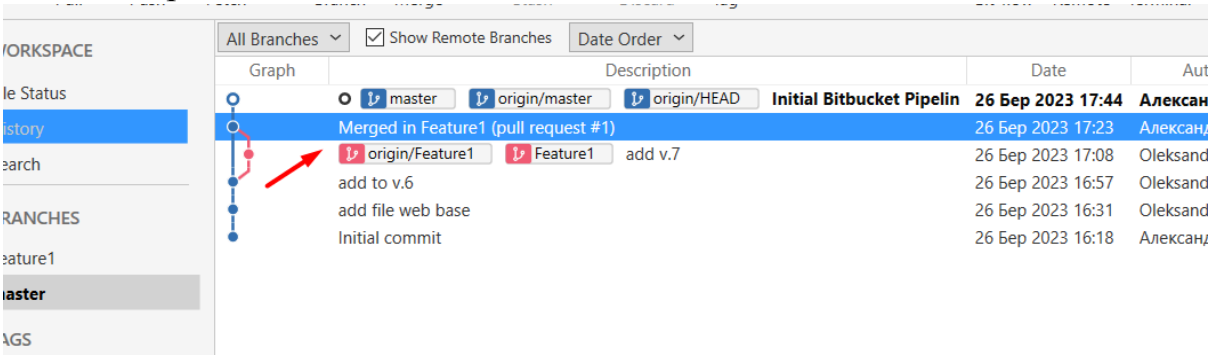
Далі переходимо до функції merge. Натискаємо.

Довідково. Є різні види злиття. *Сквош злиття (англ. Squash Merge)* - це один з способів злиття гілок (branches) у системах контролю версій, таких як Git. При використанні сквош злиття, зміни з іншої гілки об'єднуються (зливаються) з поточною гілкою, а потім ці зміни зберігаються як один коміт (commit). Це дозволяє зберегти історію змін більш зрозумілою та легко зрозумілою, зменшуючи кількість комітів та зберігаючи чистоту історії. При сквош злитті, назва коміту може бути змінена, щоб краще відображати зрозумілу суть змін. Також, цей підхід дозволяє зменшити кількість комітів, що попадають в основну гілку репозиторію, що полегшує роботу з історією змін, особливо великих проектах. Важливо враховувати, що сквош злиття може призвести до втрати деталей, якщо відбувається злиття гілок з багатьма внесками. Тому, перед застосуванням сквош злиття, необхідно ретельно розглянути зміни та переконатися, що вони не суперечать основним принципам розробки.

Коміт злиття (англ. merge commit) - це спеціальний тип коміту, який виникає при злитті двох або більше гілок в системі контролю версій, такій як Git. При злитті гілок, Git створює новий коміт злиття, який містить зміни з обох гілок, що злиті, і вказує на обидві гілки як на батьківські коміти. Коміт злиття з'являється в історії змін і має унікальний хеш-код, який ідентифікує його в Git. Коміт злиття зазвичай використовується, коли необхідно зберегти історію змін двох або більше гілок, що злиті. Крім того, він дозволяє вирішувати конфлікти, що можуть виникнути при злитті гілок.

Довідково. Важливо мати на увазі, що злиття гілок може бути необхідним лише в окремих випадках, тому не слід злити гілки занадто часто. Крім того, необхідно ретельно перевіряти зміни перед злиттям, щоб уникнути можливих проблем у майбутньому.

1.10. Передивіться створений граф у Sourcetree, зайві гілки видаліть (на бічній панелі обираєте гілку і тиснете правою клавішею на видалити). Має виглядати приблизно так:

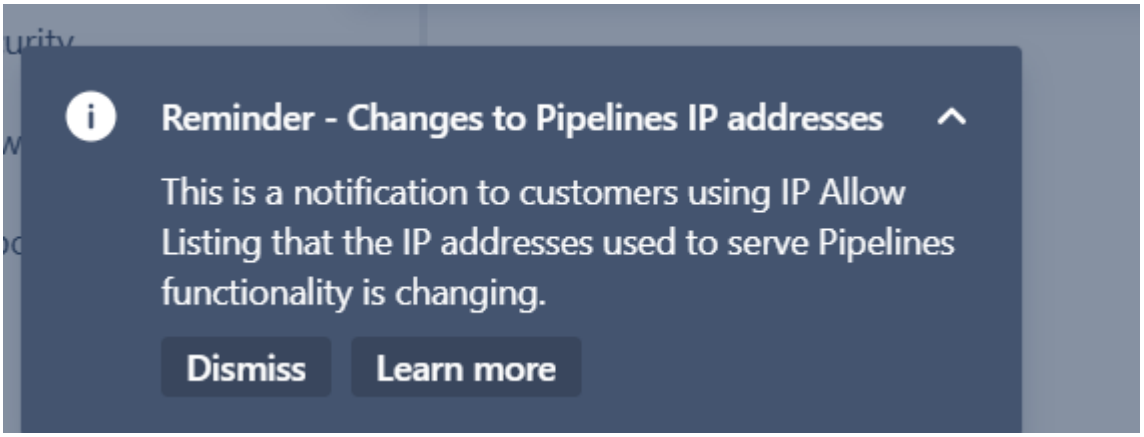


1.11. Pipelines (конвеєри) в Bitbucket. Більше інформації про Bitbucket Pipelines:

https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-pipelines/

Довідково. Bitbucket Pipelines — це інтегрована служба CI/CD, вбудована в Bitbucket. Це дозволяє автоматично створювати, тестувати та навіть розгортати свій код на основі файлу конфігурації у вашому репозиторії. По суті, це як контейнери в хмарі. Усередині цих конвеєрів можна запускати команди (як на локальній машині), але з усіма перевагами нової системи, налаштованої відповідно до ваших потреб. Для автоматичної збірки треба використовувати ідентифікатори Jira у коммітах та гілках (як ми це робили до commit fileE.txt). Є багато інструментів, які забезпечують конвеєрний підхід в налаштуванні автоматизації CI/CD, наприклад Jenkins, TeamCity, GoCD, Travis CI, CircleCI, GitHub, GitLab та ін.

Може з'явитися таке повідомлення. Як вирішити питання викладено нижче в Додатку 1.

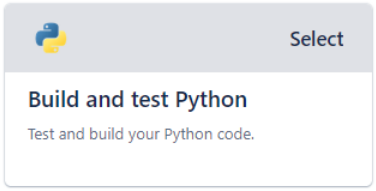


Натиснувши кнопку "Learn more", ви можете отримати більш детальну інформацію про зміни IP-адрес, включаючи список нових адрес і рекомендації щодо оновлення налаштувань.

https://www.atlassian.com/blog/bitbucket/migrating-pipelines-1-2x-steps-to-our-new-ci-cd-runtime

Обираємо рекомендований шаблон, тиснемо створити.

Довідково. Якщо ви використовували для проекту іншою мовою програмування, наприклад Python. Обираємо шаблон та дивимось параметри шаблону, можливо щось варто змінити.



Налаштування Bitbucket Pipelines з шаблоном конвеєра для Python:

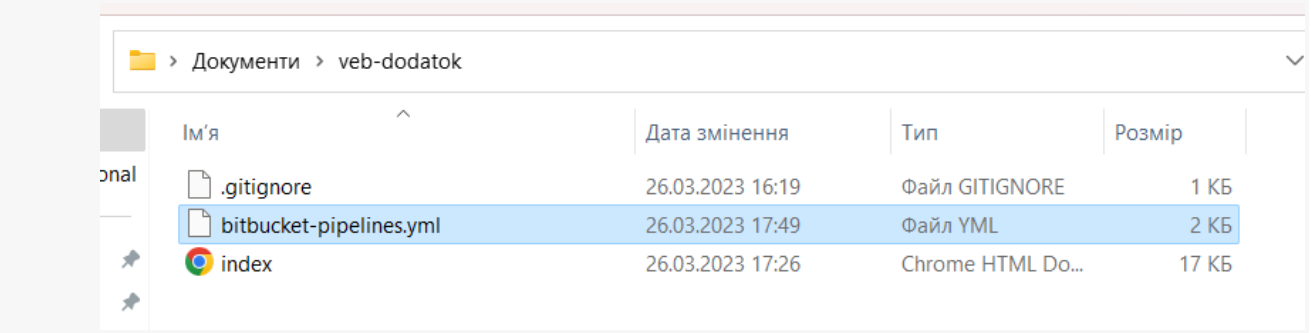
1. Виберіть "Python" у списку шаблонів конвеєрів.
2. Далі ви побачите вміст bitbucket-pipelines.yml файлу. Це файл, який містить налаштування вашого конвеєра. Файл bitbucket-pipelines.yml - це конфігураційний файл, який використовується для опису тестового процесу та автоматизованого розгортання (CI/CD) на платформі Bitbucket. Файл bitbucket-pipelines.yml дозволяє налаштувати середовище збірки, встановлювати залежності, запускати автоматичні тести та розгортати проект на відповідному середовищі.
3. У файлі bitbucket-pipelines.yml ви знайдете два етапи під назвами build та test (кожен крок виконується в новому контейнері, окремому

незалежному середовищі). Етап build відповідає за встановлення залежностей вашого проекту, а етап test відповідає за запуск тестів.

4. У розділі image ви можете вибрати базовий образ Docker, на основі якого буде створений контейнер для вашого конвеєра. За замовчуванням використовується образ python:3.7.3, але ви можете вибрати інший образ з Docker Hub. Образ Docker містить усі необхідні файли та налаштування, необхідні для запуску додатку. Цей образ можна створити самостійно або завантажити з Docker Hub, де зберігається велика кількість готових образів з операційними системами та програмним забезпеченням, яке можна використовувати для створення контейнерів.
5. Налаштування середовища - якщо ваш додаток вимагає певного середовища (наприклад, база даних або зовнішній API), вам потрібно налаштувати ці середовища для роботи в рамках вашого пайплайну.
6. Доступ до ресурсів - для запуску пайплайну вам можуть знадобитися додаткові ресурси, такі як сервери баз даних, зовнішні API тощо. Вам потрібно мати доступ до цих ресурсів або створити їх самостійно.
7. Bitbucket Pipeline дозволяє використовувати будь-які інші інструменти, які необхідні для запуску вашого додатку, тому ви можете налаштувати пайплайн так, щоб він відповідав вашим потребам.
8. У розділі pipelines ви можете налаштувати роботу вашого конвеєра. За замовчуванням, ваш конвеєр буде виконувати етапи build та test на кожен коміт у вашому репозиторії.
9. Після цього, при кожному коміті в вашому репозиторії, Bitbucket автоматично запустить ваш конвеєр та виконає всі налаштування.
10. Більше про налаштування https://support.atlassian.com/bitbucket-cloud/docs/build-test-and-deploy-with-pipelines/

1.12 Запустіть Pipelines (Run). Це займе декілька хвилин. Після запуску Pipelines відбувається автоматична компіляція, тестування та розгортання програми на вибраному середовищі (за умови налаштування).

Довідково. Файл конфігурації для Bitbucket Pipelines повинен мати назву bitbucket-pipelines.yml. Цей файл повинен бути розташований у кореневій директорії вашого репозиторію.



Bitbucket Pipelines автоматично зчитує цей файл при кожному коміті та виконує інструкції, що описані в ньому. У цьому файлі вказуються кроки, що будуть виконуватися в процесі збірки, тестування та розгортання, а також налаштування, пов'язані з розгортанням до різних середовищ (staging та production).

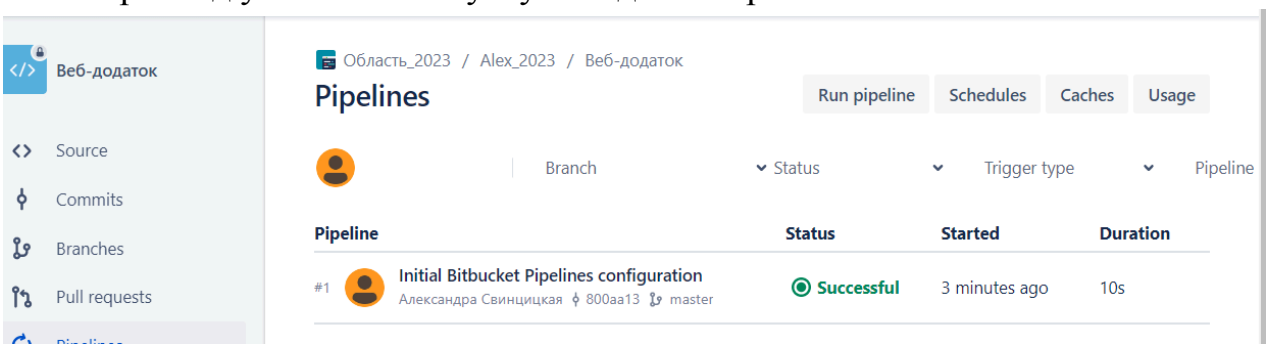
Отже, у нашому файлі передбачені такі кроки як збірка, тестування, лінтинг та сканування на безпеку, а також розгортання до **staging** та production середовищ. Крок "Deployment to Production" передбачає ручний тригер для розгортання коду в production.

Коли Pipeline завершиться, статус виконання буде відображено на сторінці вашого проекту в Bitbucket.

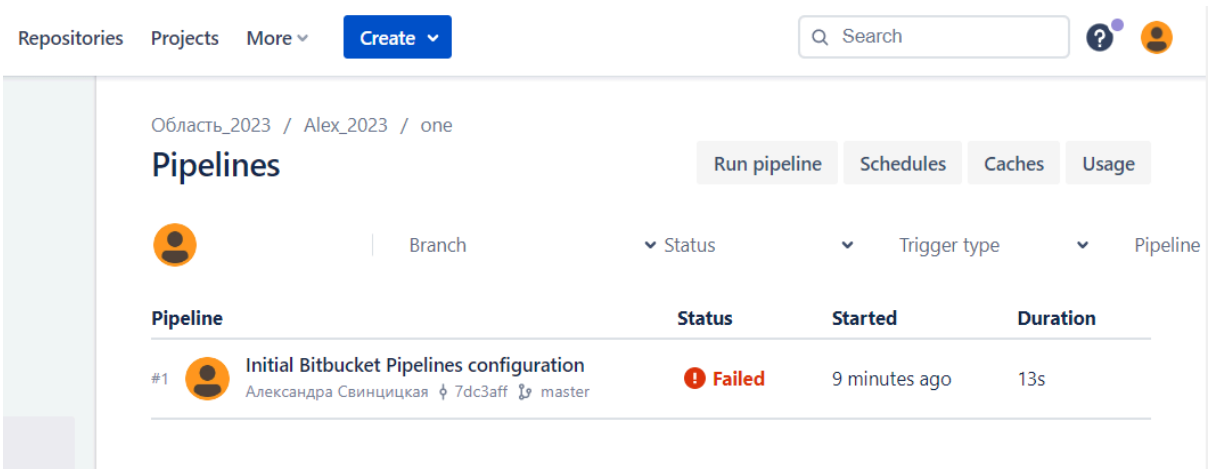
Якщо виникли проблеми, ви можете переглянути детальніші звіти про виконання кожного кроку вашого Pipeline, щоб дізнатися, де саме виникла помилка.

Успішне виконання Bitbucket Pipeline гарантує, що ваш код буде розгорнуто на вибраному середовищі без непередбачених помилок та зменшує ризик виникнення проблем при розгортанні додатку.

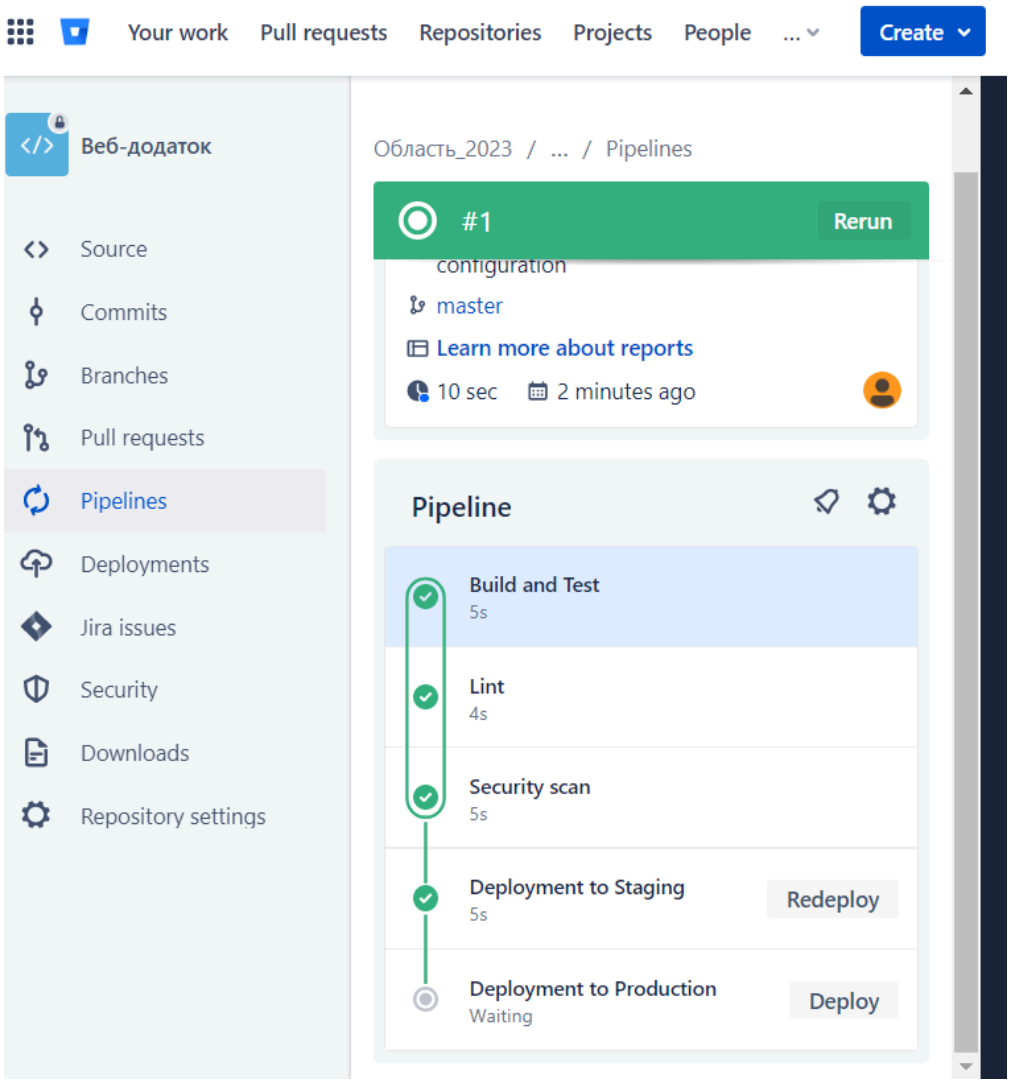
Приклад успішного запуску наведено на рис.:



або, приклад невдалого запуску наведено на рис.:



Деталі налаштованого конвеєра виглядають приблизно так:

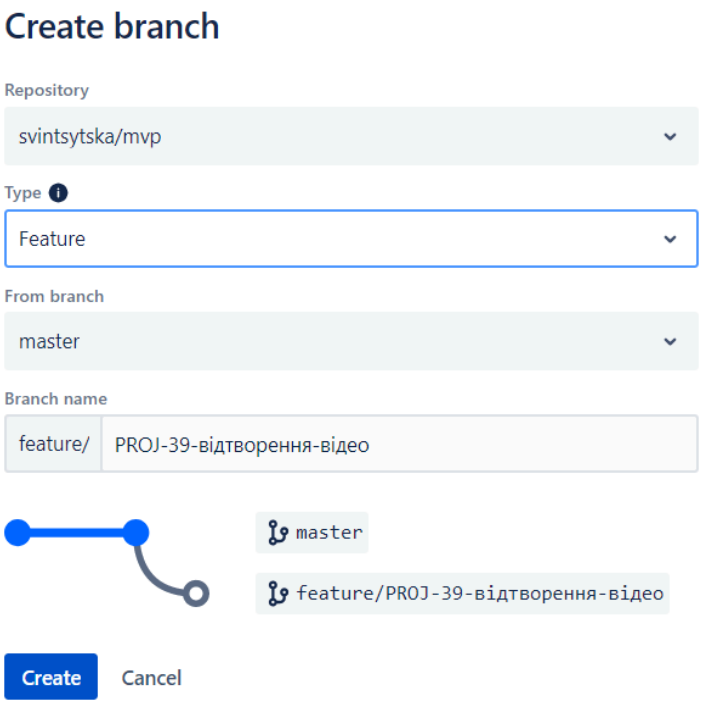


Після успішного виконання Bitbucket Pipeline, оновлення коду будуть доступні в репозиторії, і ви зможете працювати з оновленим кодом.

Довідково. Інтеграція Bitbucket Pipelines і програмного забезпечення Jira дозволяє команді автоматично відстежувати пов'язані збірки та розгортання з проблемами Jira. Для цього Ви повинні мати посилання на ключі проблем Jira у своїх комітах і гілках. Щоб побачити інформацію про збірку в Jira, просто вставте ключ проблеми в назву гілки. Наприклад, ваша гілка може називатися «feature/**ST-1**-build-mk2-boosters», і тоді інформація про збірку з'являтиметься у випуску Jira **ST-1** кожного разу, коли ваш конвеєр буде працювати з комітом в цій гілці. Щоб переглянути інформацію про розгортання в Jira, просто вставте ключ проблеми в кожне повідомлення коміту. Наприклад, ваш коміт може називатися «**DEVOPS-5** Удосконалення коду селектора», і будь-яке розгортання, яке включає це коміт, також буде представлено у випуску Jira **DEVOPS-5**. Таким чином ми автоматично виявляємо пов'язані проблеми, коли конвеєр створює та розгортає зміни коду. Після налаштування й автоматичної синхронізації інформації про збірку та розгортання з проблемами Jira, буде зручно шукати цю інформацію за допомогою мови запитів Jira (JQL).

1.13. Пройдіть весь процес в Bitbucket, починаючи від створення репозиторію і завершуючи злиття гілки. Отже:

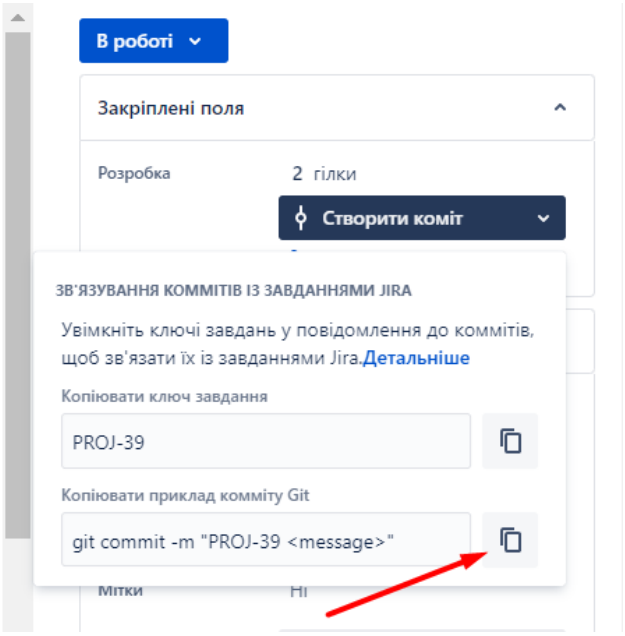
1) створюємо репозиторій в Bitbucket під назвою MVP, клонуємо його в Sourcetree.
2) переходимо в Jira (або в розділ issues на бічній панелі Bitbucket), відкриваємо картку з фічей "Відтворення відео" і в ній будемо створювати гілку Feature. Тиснемо на опцію «створення гілки» і бачимо параметри, обираємо проект і гілку, тиснемо створити, використовуючи автоматичний ідентифікатор (рис.).



Відбувається створення гілки з назвою проекту та відповідним ідентифікатором, наприклад *feature/**PROJ-39**-відтворення-відео* у віддаленому репозиторію. Тиснемо *Check out in Sourcetree* і переходимо в локальний репозиторій і погоджуємось з умовами синхронізації.

Довідково. Інтеграція Bitbucket і програмного забезпечення Jira дозволяє команді автоматично відстежувати пов’язані збірки та розгортання з проблемами Jira за допомогою ідентифікаторів проблем.

3) Далі створюємо *commit до цієї гліки* шляхом додавання *в* репозиторій *fileA.txt* з будь-яким контентом, наприклад, ***this file is used for learning 1.*** Як наслідок в нашому робочому просторі SourceTree з'являється незакомічена версія (синій виділений рядочок), закомічуємо її. Для цього на верхній панелі тиснемо «+commit» і створюємо commit з коментарем, що містить ідентифікатор проблеми, в нашому прикладі, це **git commit -m "PROJ-39 <new>".** Попереднього в Jira з робочої картки копіюємо цей параметр *commit* (рис.) та ставимо галочку push.



4) Тиснемо push і перевіряємо чи в Bitbucket є всі створені комміти і гілка.
5) Далі здійснюємо Pull requests і завершуємо роботу з цим проектом.

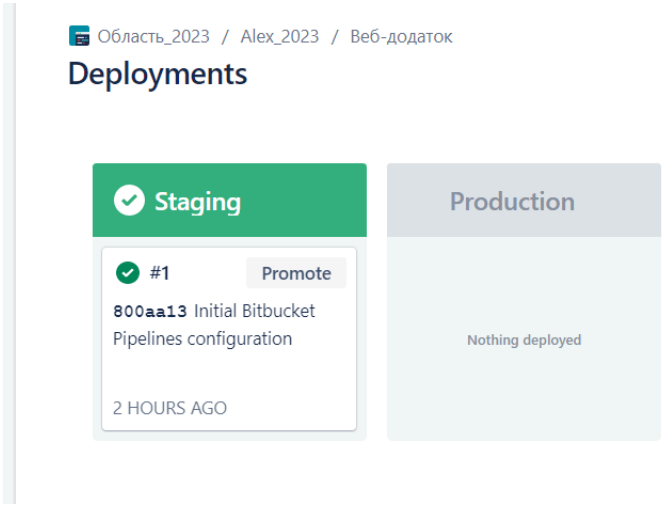
Переходимо до картки проблеми і бачимо, що з’явилась одна гілка та один комміт до цієї задачі.

1.14. Deployment (розгортання) за допомогою Bitbucket. Більше інформації: https://support.atlassian.com/bitbucket-cloud/docs/deployments/

Довідково. Основним завданням Deployment є автоматичне розгортання програми на сервері після успішного проходження тестів в Bitbucket Pipelines. За допомогою цієї опції можна проводити відстеження розгортання, щоб переглядати статус розгортань і ключову інформацію про кожну commit, яку ви розгортаєте у своїх середовищах.

Після того, як ви налаштували Bitbucket Pipelines, ви можете запустити Deployment. Розгортання в різні середовища здійснюється за допомогою конвеєрів, які можна налаштувати. Безкоштовний план включає 50 хвилин збірки на місяць.

Отже, бачимо розгортання до **staging.**

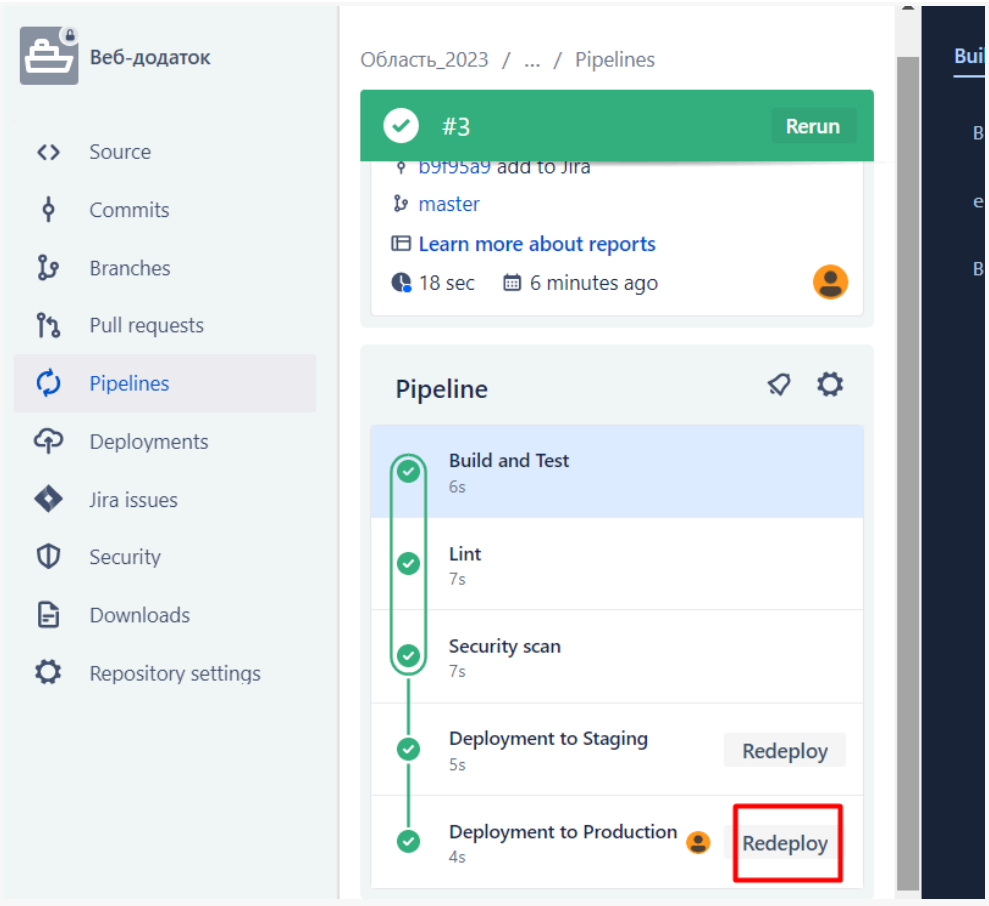


Довідково. Staging - це назва середовища, яке використовується для тестування та перевірки нової версії програмного забезпечення перед тим, як вона буде розгорнута на живому (production) сервері. Це іноді називається Pre-Production (передпродакшн) середовищем.

Середовище Staging може бути сконфігуроване як окремий сервер або окремий екземпляр хмарної інфраструктури. Він має бути якомога більш схожим на продакшн-середовище з точки зору конфігурації, але зазвичай з меншим обсягом ресурсів та об'ємом даних.

Середовище Staging може бути використане для перевірки функціональності, продуктивності, безпеки та інших аспектів програмного забезпечення перед його випуском в продакшн. Якщо тестування проходить успішно, програмне забезпечення може бути розгорнуто на продакшн-серверах.

Якщо ваш проект знаходиться на Bitbucket, то вам можна скористатися будь-якою платформою для розгортання вашого додатку.



Критерії прийняття:

Показати репозиторій «Веб додаток»

Показати створену гілку в Sourcetree

Показати Pull requests в Bitbucket

Показати Pipeline проекту Bitbucket

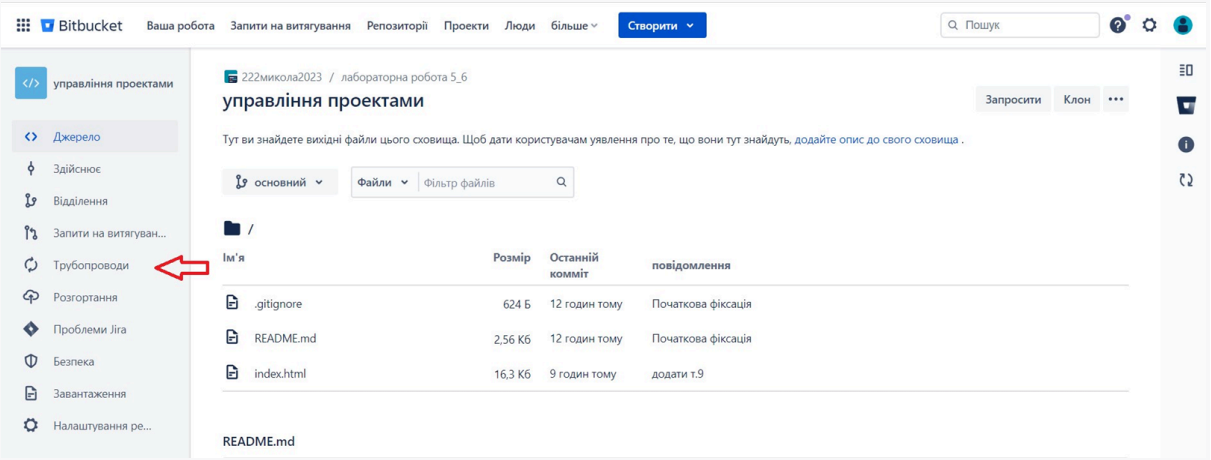
Показати в Jira картку з фічей “Відтворення відео” і в ній створену гілку Feature та Pull requests

Показати стан Deployment

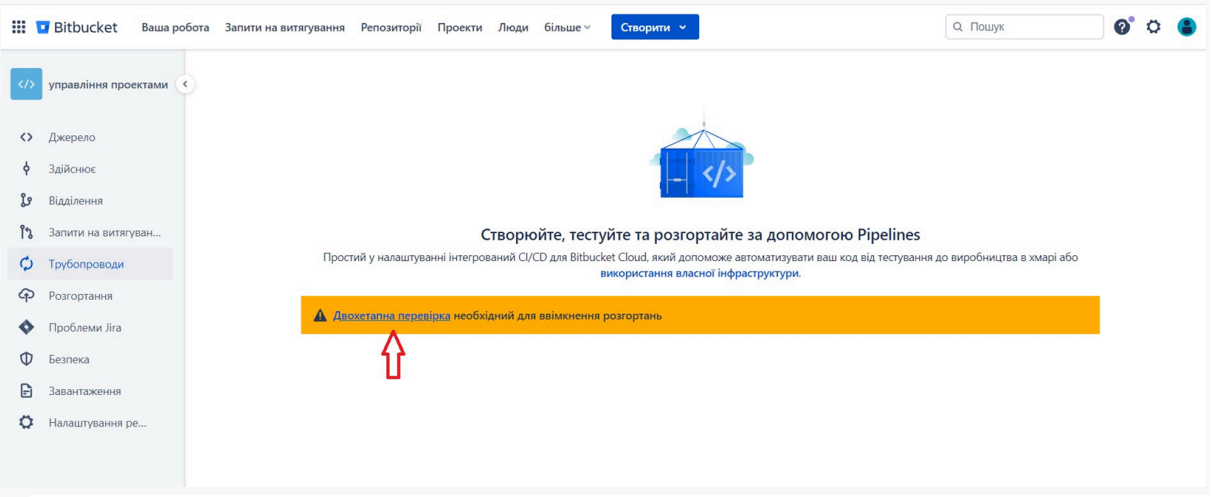
Додаток А

Довідкового!!! Якщо Pipeline потребує додаткового налаштуванні в становлення

Приступивши до виконання п. 1.11. (Pipelines (конвеєри) в Bitbucket. Більше інформації про Bitbucket Pipelines) лабораторної роботи спочатку ми переходимо до перегляду у Bitbucket створеного нами репозиторію. Натискаючи на бічній панелі кнопку «Pipelines» (трубопроводи) ми переходимо до даного розділу (наступна сторінка даного документу):

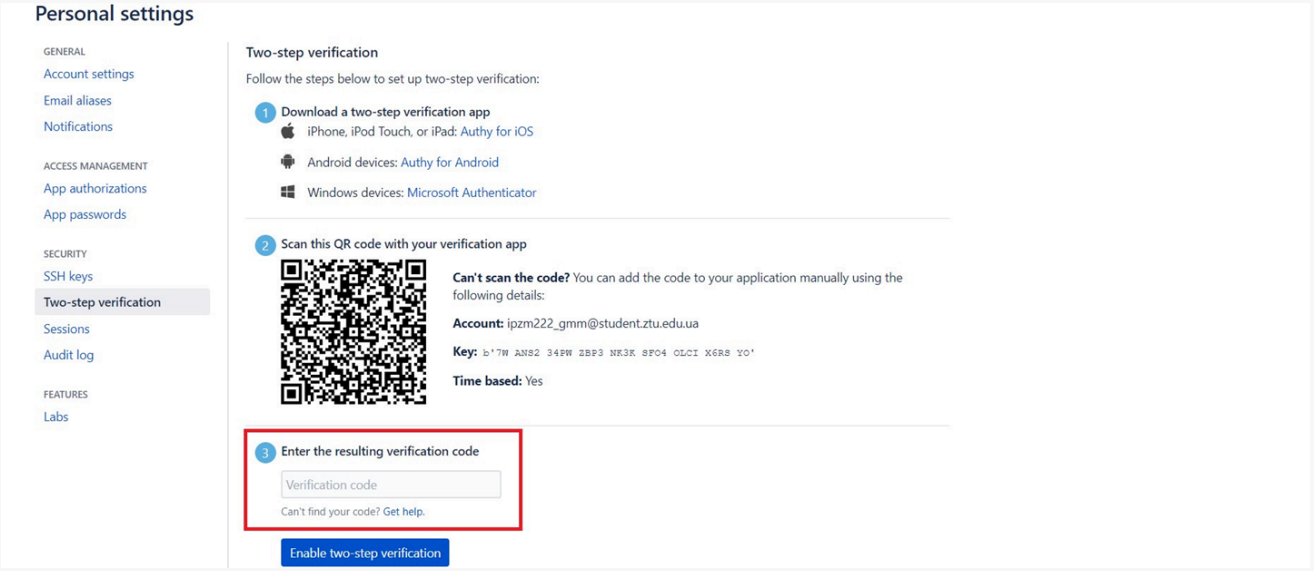


Потрапивши до розділу «Pipelines» (трубопроводи) ми бачимо, що для початку роботи з Pipelines нам потрібно пройти двоетапну перевірку. Тому натискаємо на активний напис «Двоетапна перевірка» розташований посередині екрану на жовтому фоні та потрапляємо до сторінки двоетапної перевірки (наступна сторінка даного документу):

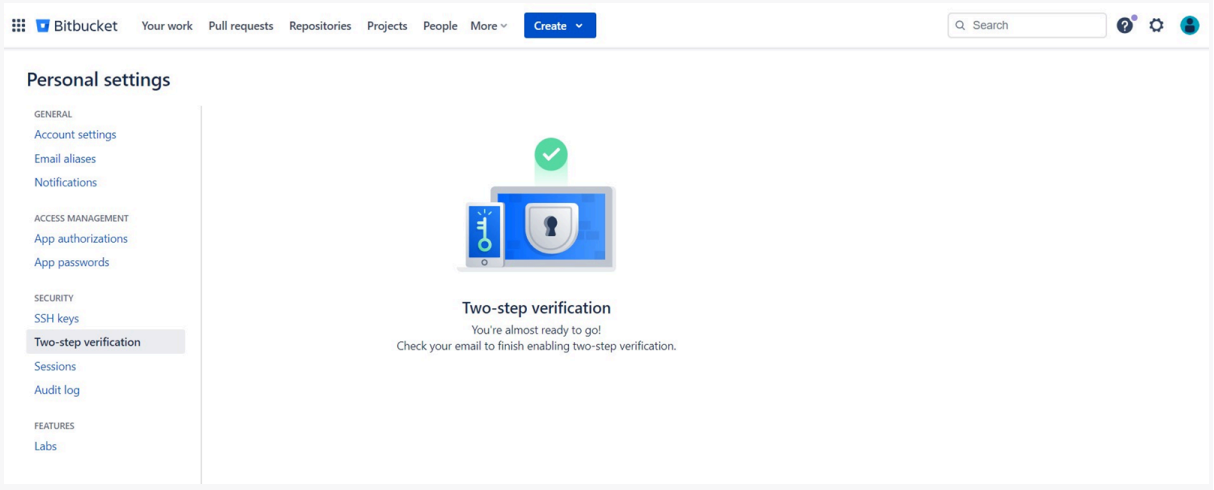


Для проходження двоетапної перевірки власникам пристроїв, які працюють на Android пропонується встановити застосунок Authy for Android (з Google Play).

Його використання показує, що він є досить незручним (запитувані користувачем SMS із кодом, застосунок не надсилає, а коли на запит користувача застосунок здійснює виклик, інтервал між повідомленням ботом цифри коду та її введенням досить не великій, помилкове ж введення коду призводить до бану на деякий час) тому краще використовувати застосунок Authenticator 7 (з Google Play). Після встановлення застосунку та запуску вбудованого сканеру камеру телефону потрібно зорієнтувати на згенерований Bitbacket QR-код (аналогічний зображеному на скріншоті розташованому на даній сторінці). Після зчитування QR-коду застосунок генерує шестизначний код, який потрібно оперативно ввести у поле 3 та натиснути клавішу Enter. Слід звернути увагу, що код швидко втрачає свою валідність. Після успішного введення коду перший етап завершується та на екран виводиться повідомлення про перехід до другого етапу перевірки (наступна сторінка даного документу) [КРИТИЧНОЮ УМОВОЮ для успішного проходження першого етапу перевірки є наявність на пристрої Android акаунту на поштову скриньку, яку було використано при реєстрації у Atlassian]: (іконка застосунку Authenticator 7)



Як зазначено на даному скріншоті користувачу для завершення другого етапу перевірки пропонується перейти до поштової скриньки, яку було використано при реєстрації у Atlassian. Перейшовши до зазначеної скриньки ми побачимо лист від Bitbacket (наступна сторінка даного документу):



У зазначеному листі повідомляється, що для завершення другого етапу перевірки потрібно натиснути на синю кнопку «Enable two-step verification». Натиснувши на дану кнопку ми повертаємось до Bitbacket, який інформує нас про успішне завершення двоетапної перевірки (наступна сторінка даного документу):

Після успішного завершення двоетапної перевірки нам потрібно перейти до нашого репозиторія та на бічній панелі натиснути кнопку «Налаштування репозиторію» (Repositiry settings) (наступна сторінка даного документу):

Натиснувши на зазначену кнопку ми потрапляємо до розділу налаштувань репозиторію (наступна сторінка даного документу):

У даному розділі нам потрібно здійснити прокрутку меню розташованого на бічній панелі до розділу «PIPELINES» та натиснувши на кнопку «Settings» перейти до налаштувань Pipelines (наступна сторінка даного документу):

Переміщуючи праворуч повзунок, розташований у центрі екрану, ми вмикаємо Pipelines (наступна сторінка даного документу):

Увімкнувши Pipelines ми можемо переходити до конфігурації Pipelines (наступна сторінка даного документу):

Як зазначено на даному скріншоті, для конфігурації Pipelines потрібно проскролити сторінку до низу або натиснути на активний напис «Create your first pipeline», який розташовано у центрі екрану:

Тепер обираємо шаблон та слідуємо за вказівками відповідної лабораторної роботи. Новачкам рекомендую обирати шаблон «Starter pipelines», який дозволить вам створити Pipeline та здійснити deploy не витрачаючи зайвого часу на конфігурацію у якій ви поки що не розумієтесь. Шаблон «Starter pipelines» швидше імітує створення Pipeline ніж виконує конкретні скрипти.