Практична робота 2. Спільна розробка проектів. Керування версіями

Теоретичні відомості

Системи керування версіями (Version Control System, VCS) – програмні інструменти, які допомагають розробникам керувати змінами в програмному коді з плином часу. Система відстежує всі зміни, які вносяться в код та зберігає їх в спеціальній базі даних. Кожна зміна зберігається як версія. При виявленні помилки розробник може повернутися назад і виконати порівняння з попередніми версіями коду та, за необхідністю, відмінити зміни. Також система дозволяє без побоювань експериментувати з кодом та зберігати найбільш вдалий результат.

Особливо такі системи є корисними при спільній роботі над проектом в команді. Наприклад, один розробник може працювати над новим функціоналом, а інший працювати над кодом в іншій частині проекту. Зміни, внесені в одну частину програмного забезпечення можуть бути несумісними зі змінами, внесеними іншим розробником. Система керування версіями відстежує зміни, внесені кожним учасником команди та допомагає виявляти конфлікти при паралельній роботі.

Розробляти програмне забезпечення можна і без системи керування версіями, але такий підхід наражає проект на величезний ризик, тому програмне забезпечення для керування версіями є невід'ємною частиною повсякденної професійної практики сучасної команди розробників програмного забезпечення.

Типи систем керування версіями

- локальні
- централізовані
- розподілені

Локальна система керування версіями – це локальна база даних, розташована на особистому комп'ютері, в якій кожна зміна файлу зберігається

як версія. Кожна версія містить лише зміни, внесені до файлу з моменту його останньої версії.

Недоліком локальної системи керування версіями є складність або майже неможливість роботи над одним проектом в команді, бо необхідно працювати за одним комп'ютером.

Також, якщо щось станеться з робочим комп'ютером, то всі файли проекту будуть втрачені.

Одним із прикладів таких систем є система керування версіями RCS, яка була розроблена у 1982 році (останнє оновлення було випущено у 2020 році). RCS поставляється з Linux'ом.

Централізовані системи керування версіями мають один сервер, який містить усі версії файлів, і кілька клієнтів можуть одночасно отримувати доступ до файлів на сервері, копіювати їх на свій локальний комп'ютер або надсилати зміни на сервер. Таким чином, клієнти можуть знати, чим займаються всі інші, а адміністратори можуть контролювати, хто що може робити. Це дозволяє легко співпрацювати один з одним в команді. Протягом багатьох років це був стандарт для керування версіями.

Централізована система також має певні недоліки. Якщо сервер вийде з ладу на деякий час, то протягом цього часу ніхто не зможе співпрацювати або зберігати нові версії. Якщо жорсткий диск, на якому знаходиться центральна база даних, пошкоджується, а резервні копії не зберігаються, то втрачається абсолютно все — вся історія проекту, крім окремих версій, які розробники мають на своїх локальних комп'ютерах.

До централізованих систем контролю версій відносять системи Subversion, Concurrent Versions System, Azure DevOps Server від Microsoft.

Розподілені системи керування версіями використовуються для подолання недоліків централізованої системи контролю версій. Клієнти повністю клонують сховище файлів проекту (репозиторій), включаючи його

повну історію. Всі копії є рівноправними і можуть синхронізуватися між собою. Якщо сервер вийде з ладу, будь-яке із клієнтських сховищ можна скопіювати на сервер, що допоможе його відновити. Кожен клон є повною резервною копією всіх даних.

До цього виду систем керування версіями відносяться Mercurial та Git. Остання є найбільш популярною системою керування версіями.

Система Git була розроблена в 2005 Лінусом Торвальдсом – розробником ядра операційної системи Linux. Система використовується безліччю професійних розробників програмного забезпечення. Вона чудово працює під управлінням різних операційних систем і може застосовуватися з безліччю інтегрованих середовищ розробки (IDE). Програма є безкоштовною і випущена під ліцензією GNU GPL версії 2.

Майже всі операції з системою керування версіями відбуваються тільки з локальним репозиторієм на комп'ютері, де встановлена система. Для резервного зберігання версій проекту та спільної роботи в команді створюється віддалений репозиторій, який за необхідністю синхронізується з локальним. Послуги віддаленого зберігання репозиторіїв надають спеціальні веб-сервіси.

Сервіси віддаленого зберігання репозиторіїв

Найпопулярнішими сервісами віддаленого зберігання репозиторіїв є веб-сервіси GitHub, GitLab та Bitbucket.

GitHub – найбільше сховище коду, яке використовується організаціями як для приватного доступу, так і для спільної роботи з відкритим кодом.

Цей сервіс був заснований у 2008 році американськими програмістами Крісом Ванстрасом (Chris Wanstrath), Томом Престон-Вернером (Thomas Preston-Werner) та Пі Джей Хайеттом (PJ Hyett). У 2018 році компанію придбала корпорація Microsoft.

Спочатку проект передбачався як спільнота розробки відкритого вихідного коду, що використовує систему керування версіями Git. Засновники GitHub із самого початку позиціонували свій проект як соціальну мережу для

програмістів: сервіс дозволив не лише публікувати власний код, а й коментувати чужі розробки, підписуватися на інших учасників та отримувати повідомлення.

У GitHub можна скопіювати будь-який опублікований репозиторій у свій профіль, щоб його модифікувати. Потім розробник може поділитися змінами із власником репозиторію за допомогою запиту на включення. Якщо власнику подобаються зміни, він може злити їх із початковим репозиторієм. Таким чином, підхід GitHub дозволив будь-якому зареєстрованому користувачу ділитися, покращувати або розвивати відкриті проекти.

На даний момент безкоштовна версія GitHub дозволяє розробникам працювати з публічними та приватними репозиторіями, робити внесок у них та співпрацювати. GitHub також має підтримку wiki – вбудований інструмент для створення та спільного використання документації версій, вбудований сервіс для створення та розміщення сайтів, підсвічування синтаксису більше ніж 200 мов програмування та безліч доповнень для підвищення продуктивності розробки та покращення співпраці.

Згідно з даними компанії, GitHub користуються більше 50 мільйонів користувачів. Ним користуються Microsoft, Facebook, Twitter, Google та інші великі корпорації. За популярністю цей сервіс є лідером.

GitLab – була заснована в 2011 році як альтернатива GitHub і BitBucket. Його головною перевагою є широка функціональність, яка поєднується з чудовим інтерфейсом користувача.

Її заснував українець Дмитро Запорожець 2011 року. Харківський програміст тоді працював в IT-компанії Sphere Software, де часто використовував сервіс GitHub. Він не влаштовував Запорожця як із робочих причин, так і в плані вартості. Так у нього з’явилася ідея створити свій аналог.

Спочатку Запорожець працював над проектом вечорами і на вихідних, але 2013-го до нього приєднався голландський підприємець Сід Сібранджі. Зараз він відповідає в компанії за бізнес-складову (CEO), а Запорожець – за технічну частину (CTO).

Однією з відмінностей сервісу є можливість встановлення системи на власних серверах. Також, серед розробників існує думка. що GitLab має більш широкі можливості по роботі з CI/CD порівняно з GitHub. CI/CD – це технологія автоматизації збірки, тестування та доставки нових версій додатків та сервісів кінцевому користувачу.

За офіційними даними компанії, систему використовують від 200 000 до 500 000 організацій, включаючи IBM, китайського гіганта електронної комерції Alibaba, японську Sony, Юліхський дослідницький центр, NASA, CERN, Invincea, видавництво O'Reilly, Обчислювальний центр Лейбніца (LRZ) і фонд GNOME, KDE.

Bitbucket – сервіс був запущений у 2008 році. Його створила австралійська команда, а пізніше придбала Atlassian у 2010 році. Спочатку сервіс підтримував тільки роботу з системою керування версіями Mercurial, однак у 2011 році почав підтримувати і систему Git.

Головною перевагою Bitbucket є можливість розмістити необмежену кількість приватних сховищ для невеликих команд (1-5 користувачів). Однак спочатку інтерфейс користувача Bitbucket був не таким простим, як у GitHub, і функціональність була недостатньо розроблена. Однак зараз ці два сервіси стають все більш схожими – і популярність BitBucket зростає.

Основною відмінністю цієї платформи є її інтеграція з Jira – одним з найбільш популярних інструментів для керування проектами в сфері розробки програмного забезпечення. В результаті розробники можуть легко відстежувати діяльність BitBucket в Jira, вимірювати їх продуктивність і підключати організаційні ради до спільних репозиторіїв. Це зручно, оскільки менеджери проекту можуть легко відстежувати організаційні та технологічні аспекти проекту.

Також, як і GitLab, сервіс можна розгорнути на власному сервері.

BitBucket не публікує регулярну статистику кількості користувачів. За їхніми офіційними новинами, у 2019 році продуктом скористалися 10 мільйонів розробників.

Хід роботи

1. Завантажте https://notepad-plus-plus.org/downloads/ та встановіть редактор Notepad++, якщо він ще не встановлений у вас в системі.
2. Завантажте та інсталюйте Git https://git-scm.com/downloads. При інсталяції залишайте усі опції за замовченням, окрім редактору, виберіть Notepad++.
3. У одній із директорій створіть папку «Project».
4. Перейдіть до папки і через контекстне меню виберіть Git Bash Here
5. В консолі введіть команду перегляду конфігурації: git config –list
6. У переліку конфігурацій немає зареєстрованого користувача. Використовуючи команди добавте свого користувача та пошту.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

7. Після цього перевірте, що вони добавлені через команду git config ---list
8. Використовуючи команду git init ініціалізуйте репозиторій.
9. Зробіть налаштування провідника, щоб відображилися приховані файли та директорії.
10. Перевірте, що в робочій папці створилася папка з назвою .git
11. Використовуючи Notepad++ у папці створіть текстовий документ «file1X» (де X – номер варіанту) з трьома рядками і збережіть.
12. Використовуючи git status перевірте стан репозиторію. Зробіть копію екрану.
13. Використовуючи команду «Get Gui Here» контекстного меню в папці проекту викличте графічний інтерфейс.
14. Проконтролюйте щоб в налаштуваннях проекту стояло кодування utf-8.
15. Проаналізуйте зміст. Зробіть копію екрану.

16. Запустіть команду для добавлення файлу на індексування, повторіть пункти 11 - 15.
17. Запустіть команду коміту з повідомленням (опція -m) «Перша версія проекту», повторіть пункти 11 - 15.
18. Порівняйте збережені копії екранів, зробіть висновок, щодо їх змісту.
19. Створіть новий файл в робочій директорії з назвою file2X.txt.
20. Запишіть туди три довільні рядки.
21. У першому файлі видаліть другий рядок, та добавте в кінець рядок з написом «четвертий рядок».
22. Додайте обидва файли до індексу та зробіть коміт:

```
git add *.txt
git commit -m 'Друга версія проекту'
```

23. Виконайте команду для перегляду історії проекту git log. Зробіть копію екрану.
24. Відкрийте графічний інтерфейс, викличте меню «Repository->Visualize master’s History», передивіться історію комітів. Зробіть копію екрану.