

КПІ ім. Ігоря Сікорського
Інститут прикладного системного аналізу
Кафедра Системного проектування

Лабораторна робота №1
«Системи контролю версій SVN, GIT»
з дисципліни «Проектування інформаційних систем»

Виконав:
Студент групи ДА-72
ННК «ІПСА»
Д'яконов Д.К.
Варіант № 5

Мета роботи: за допомогою системи контролю версій завантажити коди програми у репозиторій. Відтворити типовий цикл розробки програмного забезпечення з використанням системи контролю версій.

Задача:

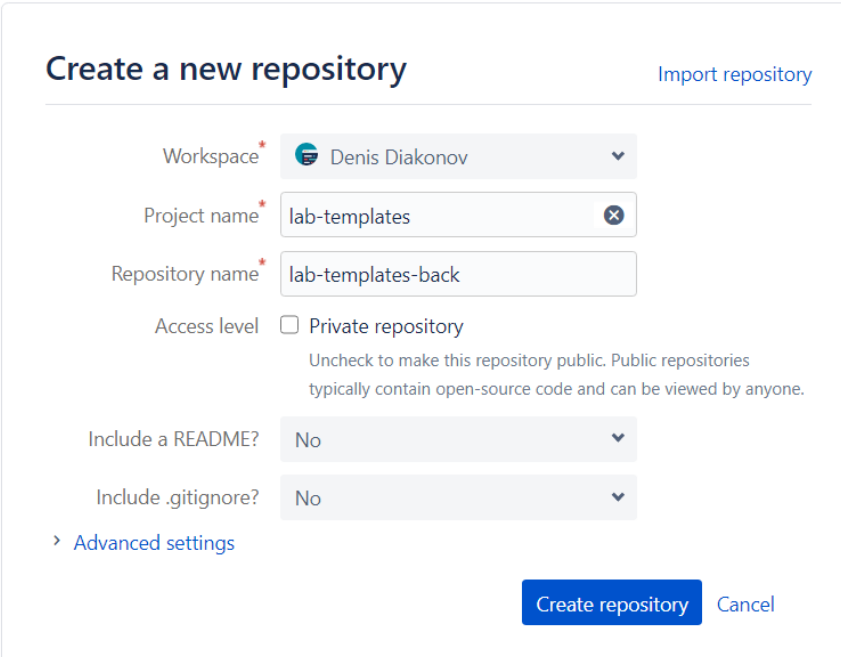
1. Вивчити основні команди роботи з репозиторіями.
2. Завантажити код програми у репозиторій.
3. Показати основний цикл роботи з програмним кодом за допомогою системи контролю версій.

Завдання:

1. Обрати безкоштовну систему репозиторія для системи контролю версіями, наприклад projectlocker, або інш.
2. Встановити клієнтське безкоштовне програмне забезпечення для роботи с системою контролю версій (GIT, SVN clients).
3. Протягом роботи над лабораторними роботами 2-6 використовувати систему контролю версіями.
4. Описати цикл розробки програмного забезпечення з використанням системи контролю версій.

Хід роботи:

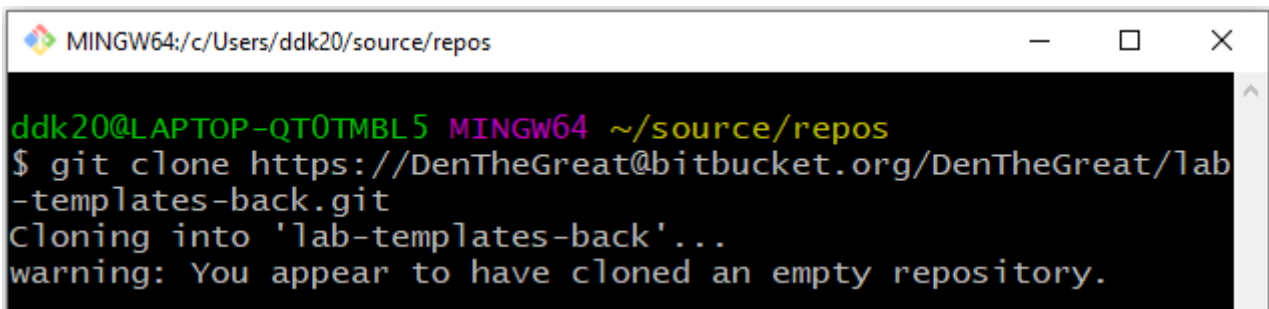
1. Створимо репозиторій на хостингу проектів Bitbucket, що працює з системою контролю версій git.



The screenshot shows the 'Create a new repository' form in Bitbucket. At the top right is a link 'Import repository'. The form contains the following fields and options:

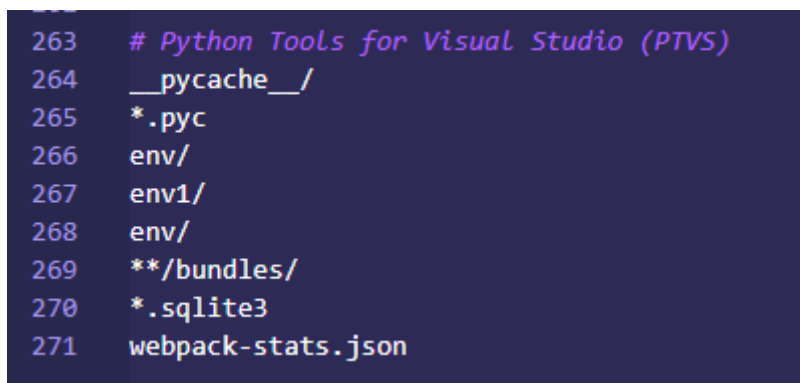
- Workspace:** A dropdown menu showing 'Denis Diakonov'.
- Project name:** A text input field containing 'lab-templates'.
- Repository name:** A text input field containing 'lab-templates-back'.
- Access level:** A checkbox labeled 'Private repository' which is currently unchecked. Below it, a note reads: 'Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.'
- Include a README?:** A dropdown menu with 'No' selected.
- Include .gitignore?:** A dropdown menu with 'No' selected.
- Below the dropdowns is a link: '> Advanced settings'.
- At the bottom right are two buttons: 'Create repository' (in blue) and 'Cancel'.

2. Склонуюмо пустий репозиторій на робочий комп'ютер використовуючи термінал git bash



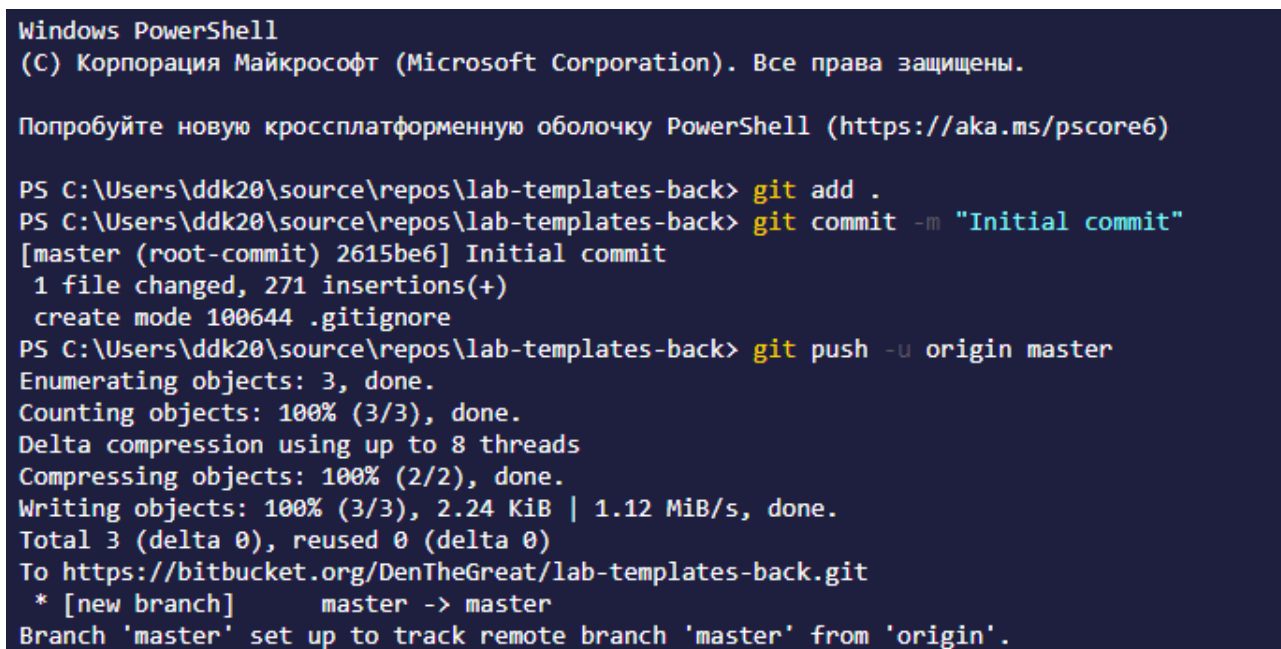
```
MINGW64:/c:/Users/ddk20/source/repos
ddk20@LAPTOP-QT0TMBL5 MINGW64 ~/source/repos
$ git clone https://DenTheGreat@bitbucket.org/DenTheGreat/lab-templates-back.git
Cloning into 'lab-templates-back'...
warning: You appear to have cloned an empty repository.
```

3. Додамо файл .gitignore, щоб сконфігурувати систему контролю версій під специфіку проекту та IDE



```
263 # Python Tools for Visual Studio (PTVS)
264 __pycache__/
265 *.pyc
266 env/
267 env1/
268 env/
269 **/bundles/
270 *.sqlite3
271 webpack-stats.json
```

4. Виконаємо перший комміт використовуючи термінал PowerShell



```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\ddk20\source\repos\lab-templates-back> git add .
PS C:\Users\ddk20\source\repos\lab-templates-back> git commit -m "Initial commit"
[master (root-commit) 2615be6] Initial commit
1 file changed, 271 insertions(+)
create mode 100644 .gitignore
PS C:\Users\ddk20\source\repos\lab-templates-back> git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 2.24 KiB | 1.12 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/DenTheGreat/lab-templates-back.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Для порівняння встановимо клієнт системи контролю версій Mercurial

1. Перейдемо у директорию з проектом та ініціалізуємо роботу системи контролю версій Mercurial

```
PS C:\Users\ddk20\source\repos\lab-templates-back> hg init .
```

Як ми бачимо, створилася системна директорія .hg

| | | |
|--------------------|------------------|--------------------|
| .git | 10.09.2020 0:14 | Папка с файлами |
| .hg | 22.09.2020 22:12 | Папка с файлами |
| accounts | 09.09.2020 23:29 | Папка с файлами |
| core | 09.09.2020 23:32 | Папка с файлами |
| env | 02.09.2020 22:15 | Папка с файлами |
| generator | 09.09.2020 23:49 | Папка с файлами |
| templates | 03.09.2020 15:22 | Папка с файлами |
| .gitignore | 23.12.2019 13:54 | Файл "GITIGNORE" |
| manage | 02.09.2020 22:21 | Python File |
| Procfile | 03.09.2020 15:33 | &Enqueue in KMP |
| README | 03.09.2020 11:56 | Файл "MD" |
| requirements | 03.09.2020 14:17 | Текстовый документ |
| requirements-local | 03.09.2020 14:17 | Текстовый документ |
| runtime | 03.09.2020 15:40 | Текстовый документ |

2. Оскільки робота розпочата у проекті на фінальних стадіях розробки, необхідно відразу додати файл з описом об'єктів, що не потребують відстеження у системі контролю версій

Основною відмінністю .hgignore від .gitignore є використання класичних регулярних виразів, на відміну від синтаксису glob, що також використовується у терміналі Bash та інших

```
.hgignore
1  __pycache__
2  env
3  .git
4  .vscode
```

3. Виконаємо перший комміт

Додамо файли проекту

```
PS C:\Users\ddk20\source\repos\lab-templates-back> hg add
adding .hgignore
adding Procfile
adding README.md
adding accounts\__init__.py
adding accounts\admin.py
adding accounts\apps.py
adding accounts\forms.py
adding accounts\migrations\0001_initial.py
adding accounts\migrations\0002_auto_20200902_2344.py
adding accounts\migrations\0003_user_variant.py
adding accounts\migrations\0004_user_gender.py
adding accounts\migrations\__init__.py
adding accounts\models.py
adding accounts\serializers.py
adding accounts\tests.py
adding accounts\views.py
adding core\__init__.py
adding core\asgi.py
adding core\settings.py
adding core\settings_prod.py
adding core.urls.py
adding core\wsgi.py
adding generator\__init__.py
adding generator\admin.py
adding generator\apps.py
adding generator\migrations\0001_initial.py
adding generator\migrations\__init__.py
adding generator\models.py
adding generator\serializers.py
adding generator\tests.py
adding generator\utils.py
adding generator\views.py
adding manage.py
adding requirements-local.txt
adding requirements.txt
adding runtime.txt
adding templates\lab.docx
```

Зафіксуємо їх як комміт

```
PS C:\Users\ddk20\source\repos\lab-templates-back> hg commit -m "Initial commit"
```

4. Порівняємо статуси систем

```
PS C:\Users\ddk20\source\repos\lab-templates-back> hg status

PS C:\Users\ddk20\source\repos\lab-templates-back> git status
On branch master
Your branch is up to date with 'origin/master'.
```

Висновок: Під час даної лабораторної роботи було створено репозиторій на платформі Bitbucket та виконано перший комміт.

При подальшій розробці план використання системи контролю версій git полягає у створенні нових коммітів при кожному закінченні робочої сесії або ж при виконанні окремо взятих задач.

Також в лабораторній роботі було використано систему контролю версій Mercurial для порівняння з системою git.

І хоча системи мають дуже схожий синтаксис команд та схему роботи в цілому, наразі Mercurial у IT-спільноті не вважається актуальною технологією, особливо після припинення підтримки системи контролю версій хостингом проектів Bitbucket у лютому 2020.

З плюсів Mercurial можна виділити спрощений синтаксис та кращу роботу з особливо великими проектами, проте всі ці плюси анулюються дуже обмеженим вибором систем для хостингу проектів та меншою кількістю можливостей, у порівнянні з git.