НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ
Кафедра мультимедійних систем

КУРСОВА РОБОТА

на тему:

«Проектування і реалізація CI/CD електронної системи»

Виконав:
Студент 4 року навчання
Бакалаврської програми
Бойцов С. В.
Керівник курсової роботи:
Корнійчук М.А.,
асистент

Київ 2021

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
доцент, кандидат наук
_____О. П. Жежерун
(підпис)
„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Бойцову С.В. факультету ФІ 4 курсу
ТЕМА Проектування і реалізація CI/CD електронної системи

Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Вступ
Огляд CI/CI процесу
Проектування і реалізації власного рішення
Висновки
Джерела
Додатки

Дата видачі „___” _____ 2021 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання курсової роботи

Тема: «CI/CD та Deploy електронної системи»

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	19.10.2020	
2.	Огляд технічної літератури за темою роботи.	02.12.2020	
3.	Огляд процесу	25.01.2021	
3.	Структурна розробка власного рішення	30.02.2021	
4.	Детальна розробка власного рішення	15.03.2021	
5.	Остаточне оформлення презентації	10.04.2021	
6.	Захист курсової роботи (проекту)	20.04.2021	

Студент _____

Керівник _____

“_____”_____

Зміст

1. Анотація

Метою роботи є дослідження, проектування та реалізація процесу Постійної інтеграції та безперервної доставки при роботі над системами що передбачають розподілену команду розробників.

Дана робота складається з 3 розділів:

У першому було розглянуто процес CI/CD загалом. У другому було оглянуто інструменти роботи з CI/CD. У третьому було спроектовано та реалізовано процес CI/CD для системи пошуку документів по Державному реєстру судових рішень.

2. Вступ

Сучасний світ вимагає бути дуже адаптивним при розробці програмного забезпечення. Планувати в довгостроковій перспективі стає неможливо, оскільки початкові умови змінюються швидше ніж відбувається процес розробки нового програмного забезпечення. Одним із вирішень даної проблеми є запровадження методологій Agile. Але крім проектних змін у роботі команд розробників для реалізації даної концепції потрібна зміна у технічному адмініструванні даного процесу.

Саме створення і вдосконалення технологій Постійної інтеграції та безперервної доставки дозволило підняти рівень роботи над цифровими продуктами. Постійна інтеграція стала неймовірно корисною та популярною за останні кілька років. Багато компаній почали використовувати його як засіб доставки коду за лічені хвилини та переконавшись, що все працює без помилок. Це значно збільшило загальний інтерес до CI / CD.

Концепція постійної інтеграції полягає в тому, щоб дати розробникам можливість постійно вводити новий код в одну систему, щоб код і система постійно оновлювалися. CI / CD - одна з найвпливовіших змін у світі бізнесу за останнє десятиліття. Можливість постійно змінювати та оновлювати свій бізнес була нечуваною лише кілька років тому. [4]

Метою роботи є дослідження, проектування та реалізація процесу Постійної інтеграції та безперервної доставки при роботі над системами що передбачають розподілену команду розробників, огляд, визначення переваг і особливостей платформ CI/CD.

Практичною реалізацією в межах даної роботи є імплементація процесу Постійної інтеграції та безперервної доставки в груповий проект над створенням пошуку документів по Державному реєстру судових рішень

Саме з цих причин темою курсової було обрано проектування і реалізацію CI/CD для електронної системи.

3. Огляд CI/CD процесу

3.1. Про CI/CD загалом

Постійна інтеграція (CI) та безперервна доставка (CD) втілюють культуру, набір принципів роботи та збір практик, які дозволяють командам розробників додатків частіше та надійніше доставляти зміни коду. Реалізація також відома як pipeline (конвеєр) CI / CD. [1]

CI (Постійна інтеграція) - це філософія кодування та набір практик, які спонукають команди розробників впроваджувати невеликі зміни та часто реєструвати код у сховищах контролю версій (git). Оскільки більшість сучасних додатків вимагають розробки коду на різних платформах та інструментах, команда потребує механізму інтеграції та перевірки своїх змін.

Технічною метою CI є створення послідовного та автоматизованого способу створення, упаковки та тестування додатків. Завдяки послідовності процесу інтеграції команди частіше здійснюють зміни коду частіше, що призводить до кращої співпраці та якості програмного забезпечення.

CD (Безперервна доставка) розпочинається там, де закінчується безперервна інтеграція. CD автоматизує доставку додатків до вибраних інфраструктурних середовищ. Більшість команд працюють із кількома середовищами, крім виробничих, таких як середовища розробки та тестування, а CD гарантує автоматичний спосіб просування змін до них.

Інструменти CI / CD допомагають зберігати специфічні для навколишнього середовища параметри, які повинні бути упаковані при кожному постачанні. Потім автоматизація CI / CD виконує будь-які необхідні виклики послуг до веб-серверів, баз даних та інших служб, які, можливо, доведеться перезапустити або дотримуватися інших процедур під час розгортання програм.

3.2. Переваги CI

Постійна інтеграція - це філософія розвитку, яка підтримується механікою процесів та деякою автоматизацією. Практикуючи CI, розробники часто вводять свій код у сховище контролю версій, і більшість команд мають мінімальний стандарт комітування щонайменше щодня. Обґрунтування цього полягає в тому, що легше виявити дефекти та інші проблеми якості програмного забезпечення на менших диференціалах коду, ніж на більших, розроблених протягом тривалого періоду часу. Крім того, коли розробники працюють на коротших циклах комітів, менша ймовірність того, що кілька розробників редагують один і той самий код і вимагають об’єднання під час фіксації. Команди, що впроваджують безперервну інтеграцію, часто починають із конфігурації контролю версій та визначення практики. Незважаючи на те, що перевірка коду проводиться часто, функції та виправлення реалізовані як на короткому, так і на більш тривалому періоді часу. Команди розробників, які практикують безперервну інтеграцію, використовують різні методи, щоб контролювати готовність функціонування та коду до виробництва.

3.3. Деталі CD

Безперервна доставка - це автоматизація, яка спрямовує програми до середовищ доставки. Більшість команд розробників, як правило, мають одне або кілька середовищ розробки та тестування, де зміни додатків проводяться для тестування та перегляду. Інструмент CI / CD, такий як Jenkins, CircleCI, AWS CodeBuild, Azure DevOps, Atlassian Bamboo або Travis CI, використовується для автоматизації кроків та забезпечення звітності.

Типовий конвеєр CD має етапи побудови, тестування та розгортання. Більш складні трубопроводи включають багато таких етапів:

- Витягування коду з контролю версій та виконання збірки.
- Виконання будь-яких необхідних етапів інфраструктури, які автоматизовані як код для відновлення або злому хмарної інфраструктури.
- Переміщення коду до цільового обчислювального середовища.
- Управління змінними середовища та налаштування їх для цільового середовища. Переміщення компонентів програми до відповідних служб, таких як веб-сервери, служби API та служби баз даних.
- Виконання будь-яких кроків, необхідних для перезапуску служб або виклику кінцевих точок служби, необхідних для нового натискання коду.
- Виконання безперервних тестів і середовищ відкату, якщо тести не проходять. Надання даних журналу та попереджень про стан доставки.

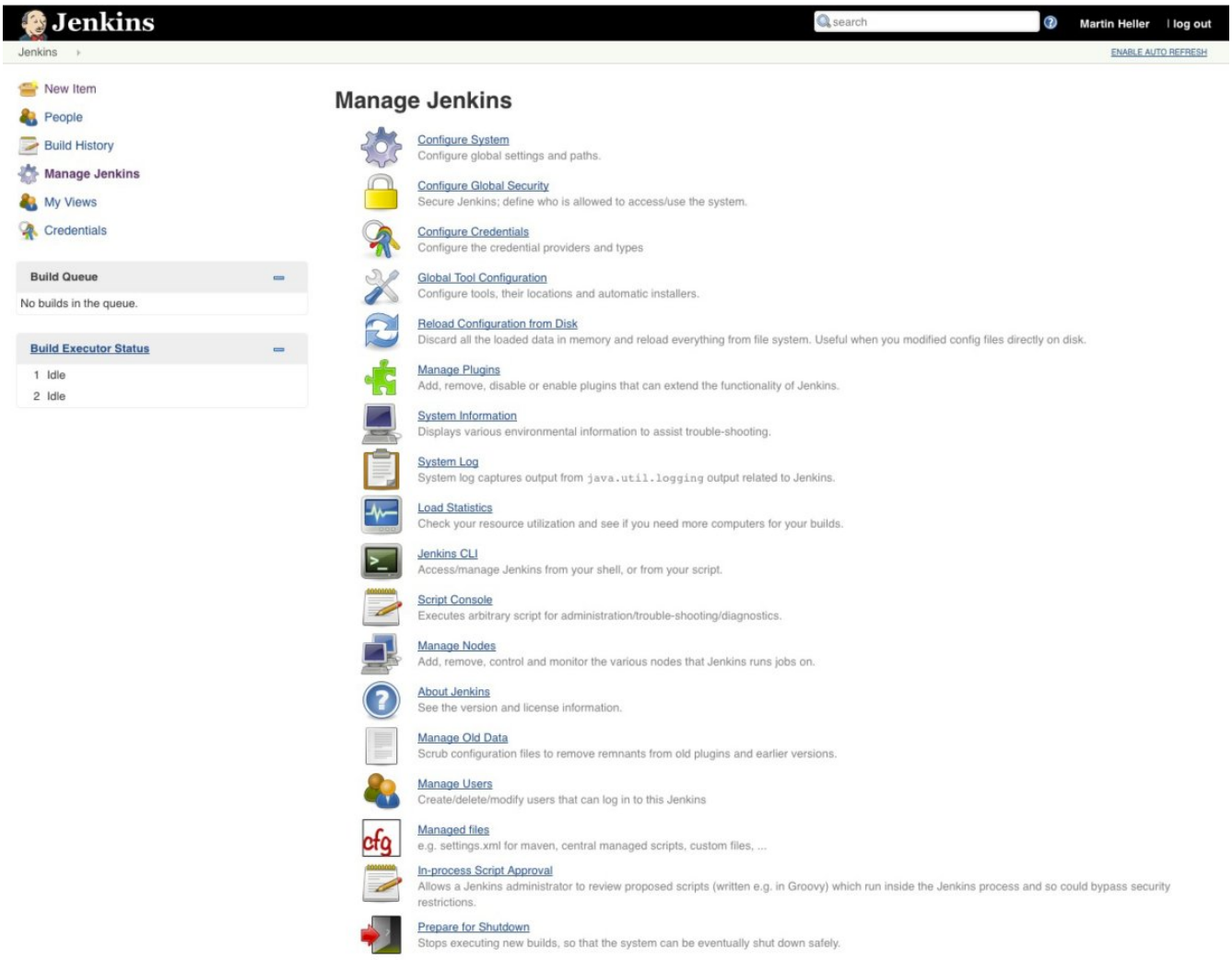
Як приклад, користувачі Jenkins визначають свої конвеєри у файлі Jenkins, який описує різні етапи, такі як побудова, тестування та розгортання. Змінні середовища, параметри, секретні ключі, сертифікації та інші параметри оголошуються у файлі та посилаються на них поетапно. Розділ повідомлення обробляє умови помилок та сповіщення. Більш складний CD може мати інші кроки, такі як виконання синхронізації даних, архівування інформаційних ресурсів або виконання виправлення програм та бібліотек. Інструменти CI / CD зазвичай

підтримують ринок плагінів. Наприклад, Дженкінс перелічує понад 1500 плагінів, які підтримують інтеграцію зі сторонніми платформами, користувальницький інтерфейс, адміністрування, управління вихідним кодом та управління збірками.

4. Огляд інструментаріїв CI/CD

4.1. Jenkins

Jenkins - це сервер автоматизації з відкритим вихідним кодом, в якому виконується центральна збірка та процес CI. Це, зокрема, програма на базі Java з доступними пакетами для Windows, macOS та інших Unix-подібних операційних систем. Завдяки сотням доступних плагінів, Дженкінс підтримує створення, розгортання та автоматизацію проектів з розробки програмного забезпечення. [3]



Зображення 1 - Інтерфейс Jenkins

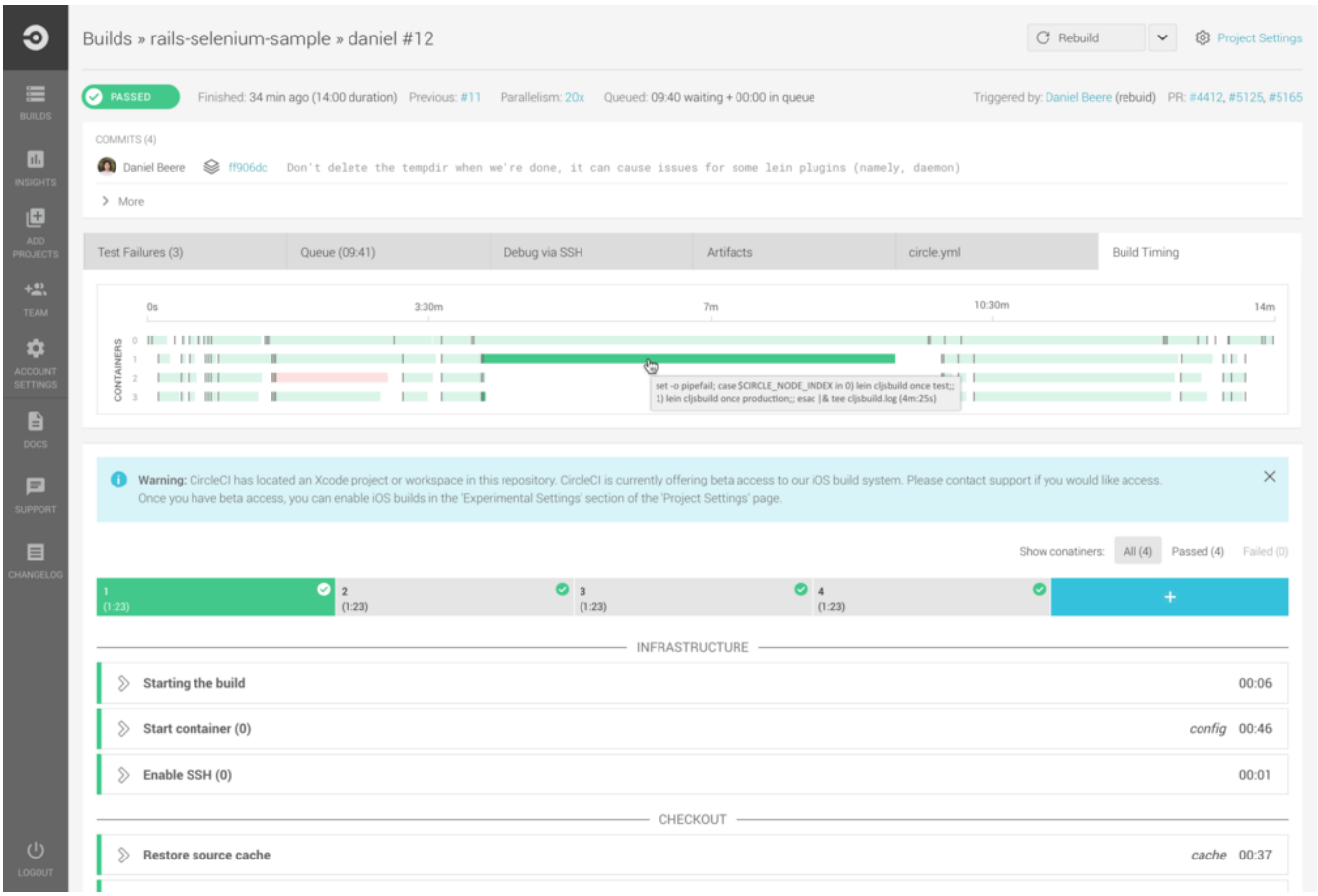
Основі фічі:

- Проста установка та оновлення до різних операційних систем
- Простий та простий у використанні інтерфейс
- Розширюється завдяки величезному ресурсу плагінів на основі спільноти
- Проста конфігурація середовища в інтерфейсі користувача
- Підтримує розподілені збірки архітектури master-slave
- Побудуйте графіки на основі фраз
- Підтримує виконання оболонок і команд Windows у кроках попередньої збірки Підтримує повідомлення про стан збірки

Ціна: повністю безкоштовно

4.2. CircleCI

CircleCI - це сучасна платформа безперервної інтеграції та безперервної доставки (CI / CD). Рішення CircleCI Enterprise можна встановити у приватній хмарі чи датацентрі і можна використовувати безкоштовно протягом обмеженого часу. CircleCI автоматизує створення, тестування та розгортання програмного забезпечення. Є можливості інтегрувати CircleCI з GitHub, GitHub Enterprise та Bitbucket, щоб створювати збірки під час комітування нових рядків коду.



Зображення 2 - Інтерфейс CircleCI

Основні фічі:

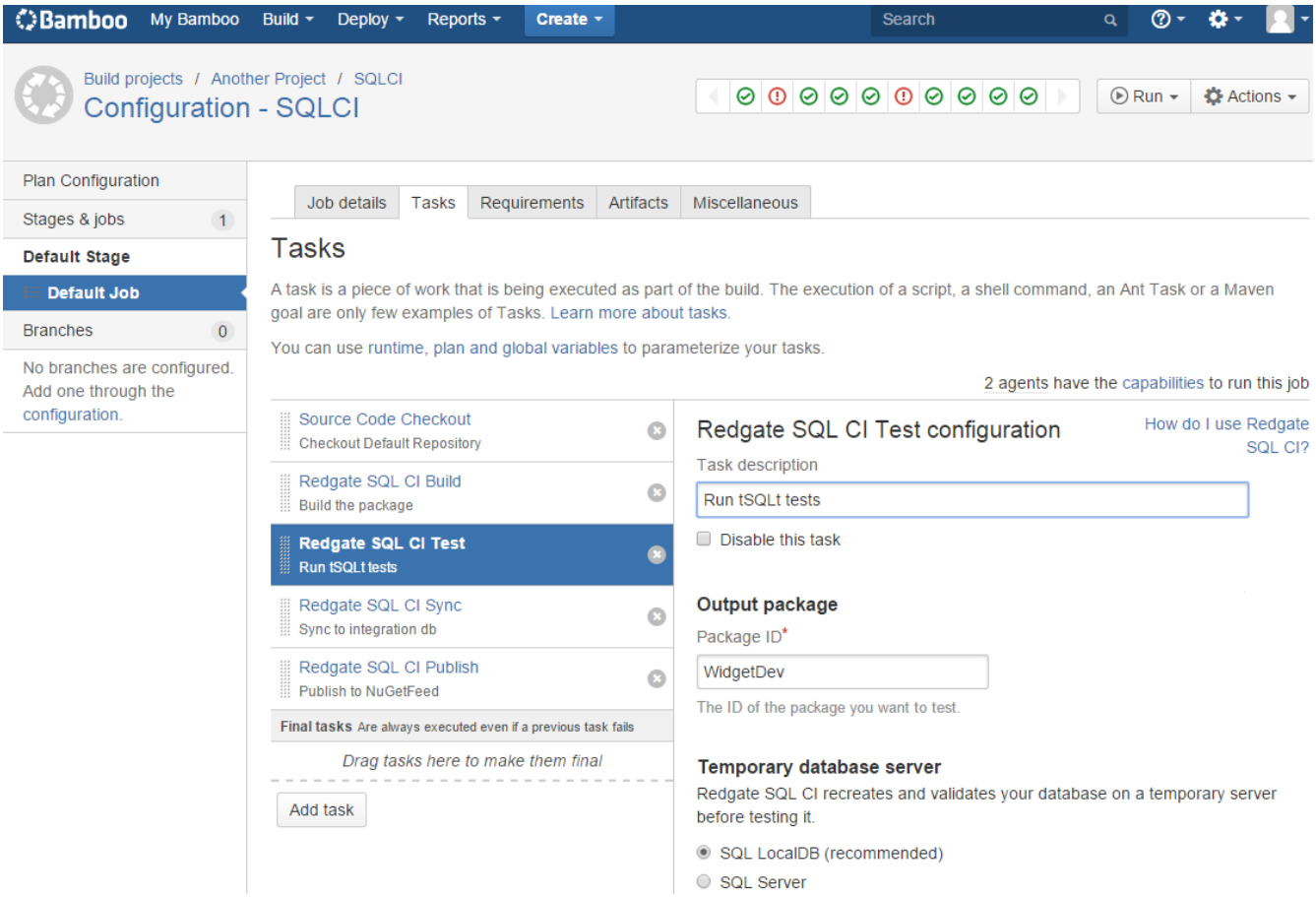
- Інтеграція з Bitbucket, GitHub та Cloud Enterprise
- Використовує контейнер або віртуальну машину для побудови
- Пряма налагодження

- Автоматизоване розпаралелювання
- Швидке тестування
- Власний текст та оновлення чату
- Розгортання є безперервним і залежить від галузі
- Цілком настроюється
- Команди автоматизованого злиття та користувацьких завантажень пакетів
- Швидке налаштування та необмежена конструкція

Ціна: Плани Linux починаються з можливості запускати одне завдання безкоштовно, без будь-якого паралелізму. Проекти з відкритим кодом отримують ще три безкоштовні контейнери. Є можливість ознайомитися з цінами для визначення того, який план потрібен під час реєстрації.

4.3. Bamboo

Bamboo - це сервер безперервної інтеграції, який автоматизує управління випуском програмних програм і, таким чином, створює конвеєр постійної доставки. Bamboo включає розробку та функціональне тестування, призначення моделей, маркування оновлень, розгортання та увімкнення нових вихідних моделей.



Зображення 3 - Інтерфейс Bamboo

Основні фічі:

- Забезпечує підтримку до 100 віддалених агентів
- Можливість паралельно запускати групи тестів та отримувати швидкий фідбек
- Створює зображення та завантажує
- Дозволи для кожного середовища, які дозволяють розробникам та тестувальникам розгортати за запитом у своїх середовищах, поки вихід

залишається заблокованим

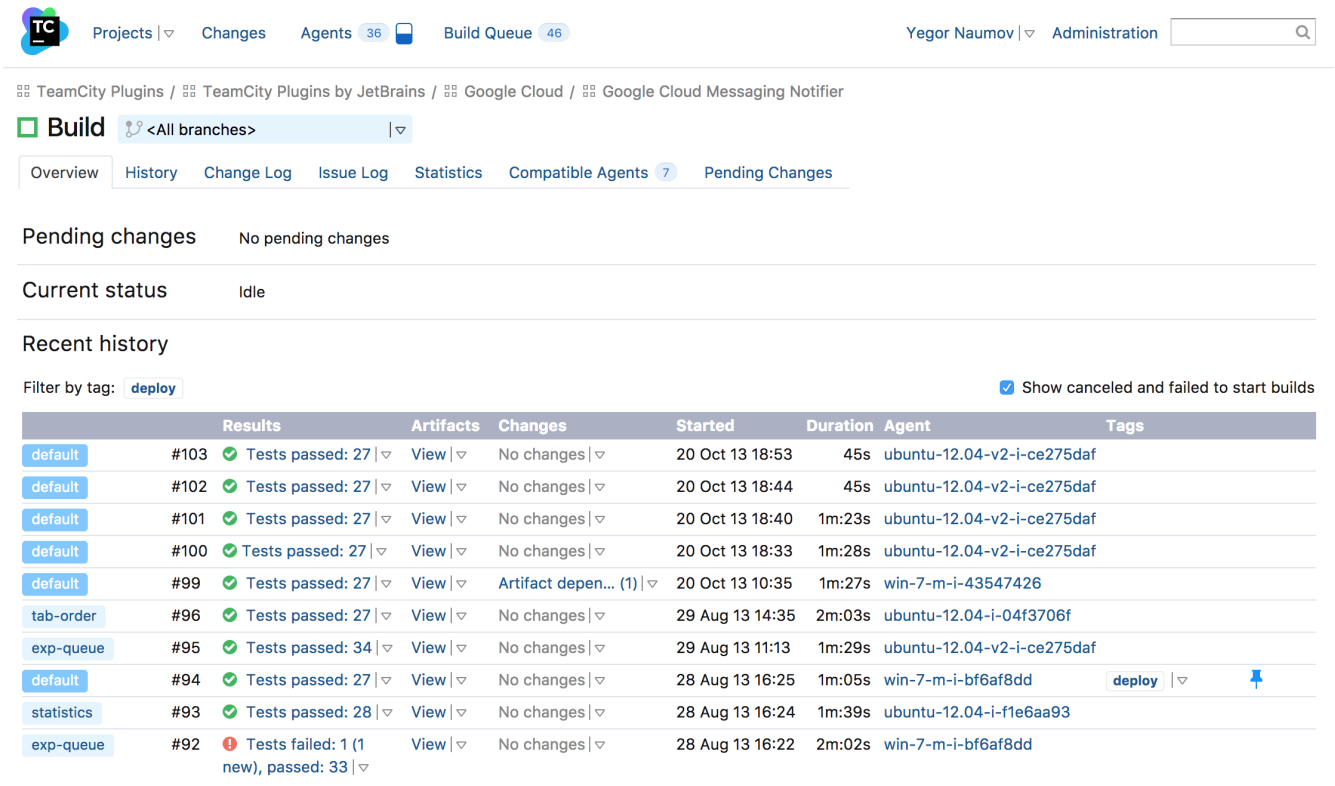
- Виявляє нові гілки в Git, Mercurial, SVN Repos і автоматично застосовує до них схему CI основної лінії
- Тригери будуються на основі модифікацій, знайдених у сховищі
- Пересилає повідомлення від Bitbucket, по заданому графіку, завершення іншої збірки або будь-якої їх комбінації

Ціни: Діапазони цін Bamboo базуються на агентах, а не на користувачах, або на "збірках slaves". Чим більше агентів вдається запустити одночасно, тим більше процесів він може запустити - або в одній збірці, або в різних збірках.

4.4. TeamCity

TeamCity - це сервер управління збіркою та постійної інтеграції для JetBrains. TeamCity - це інструмент безперервної інтеграції, який допомагає розробляти та розгортати різні типи проектів. TeamCity працює в середовищі Java та інтегрує Visual Studio та IDE. Інструмент може бути встановлений як на серверах Windows, так і на Linux і підтримує такі проекти. NET та open-stack.

TeamCity 2020.1 забезпечує умовні етапи побудови, дозволяє запускати агенти побудови в кластер Kubernetes і інтегрується з Azure DevOps та Jira Software Cloud. Він вводить більше функціональних можливостей у багатовузольну систему для вторинних серверів, постачається з новим сповіщувачем Slack і має кілька основних удосконалень експериментального інтерфейсу.



Зображення 4 - Інтерфейс TeamCity

Основні фічі:

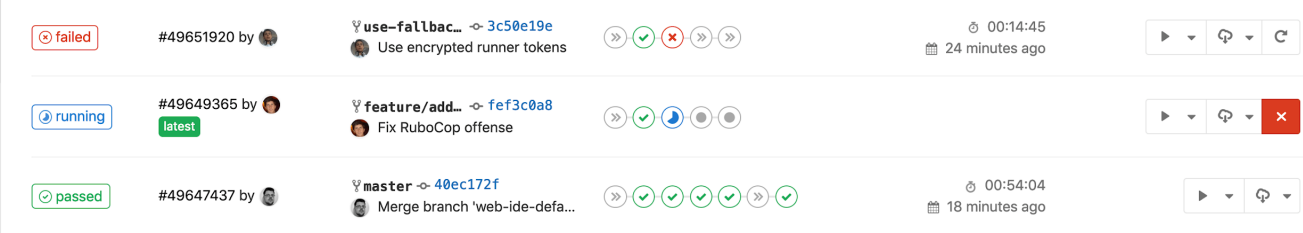
- Надає кілька способів для повторного використання в підпроекті батьківських параметрів та конфігурацій основного проекту
- Паралельний прогон працює одночасно в різних середовищах
- Дозволяє створювати історію, переглядати звіти про історію тестів, закріплювати, відмічати та додавати вибране
- Легко налаштувати, взаємодіяти та розширювати сервер
- Зберігає CI Server стабільним та функціональним
- Гнучке управління користувачами, призначення ролей користувачів, групування користувачів, різні методи автентифікації користувачів та журнал з усіма діями користувача для забезпечення прозорості всіх дій сервера

Ціна: TeamCity - комерційний інструмент, що має як безкоштовні, так і власні ліцензії.

4.5. GitLab

GitLab - це набір інструментів, призначених для обробки різних аспектів життєвого циклу створення програмного забезпечення. Основним продуктом є менеджер сховища Git, орієнтований на Інтернет, з такими функціями, як моніторинг видань, звітування та wiki.

GitLab дозволяє кожному коміту або натисканню запускати збірки, запускати тести та розгортати код. Ви можете створювати завдання на віртуальній машині, контейнері Docker або на певному сервері.



Зображення 5 - Коміти та різні ступені тестів на GitLab CI

Основні фічі:

- Інструменти розгалуження для перегляду, створення та управління кодом та даними проекту
- Можливості для планування, білду та керування кодами та даними проектів з єдиної розподіленої системи контролю версій
- Єдине джерело “правдивості” та масштабованості для співпраці над проектами та кодом
- Допомагає командам повністю охопити CI, автоматизуючи розробку, інтеграцію та перевірку вихідного коду
- Забезпечує сканування контейнерів, статичне тестування безпеки додатків (SAST), динамічне тестування безпеки додатків (DAST) та сканування

залежностей для забезпечення безпечних програм разом із дотриманням ліцензій

- Допомагає автоматизувати та скоротити випуски та доставку додатків

Ціна: GitLab - це безкоштовний комерційний пакетний інструмент. Він пропонує на місці та / або загальнодоступний хмарний хостинг SaaS на GitLab або у вашому випадку.

4.6.Порівняння

	Jenkins	CircleCI	Bamboo	TeamCity	GitLab
Відкрите ПО	Так	Ні	Ні	Ні	Ні
Складність налаштування і використання	Середня	Середня	Середня	Середня	Середня
Вбудовані фічі	3/5	4/5	4/5	4/5	4/5
Інтеграція	5/5	3/5	3/5	4/5	4/5
Хостинг	On-Premise та в хмарі	On-Premise та в хмарі	On-Premise та в хмарі	On-Premise	On-Premise та в хмарі
Безкоштовна версія	Є	Є	Є	Є	Є
Вартість ліцензування	Безкоштовно	Від 39$/міс	Від 10$	Від 299$	Від 4$/міс

5. Проектування і реалізація власного рішення

5.1. Вибір та опис інструменту

Оскільки серед найбільших CI/CD систем Jenkins є відкритим та безкоштовним ПО, з середньою складністю налаштування, з хорошим набором вбудованих фіч та достатнім рівнем інтеграції, його було обрано для подальшої роботи як інструмент для постійної інтеграції та безперервної доставки.

Jenkins пропонує простий спосіб встановити середовище безперервної інтеграції або безперервної доставки (CI / CD) для майже будь-якої комбінації мов та сховищ вихідних кодів, що використовують конвеєри, а також автоматизує інші рутинні завдання розробки. Незважаючи на те, що Jenkins не усуває необхідності створювати сценарії для окремих кроків, він дає швидший та надійніший спосіб інтегрувати весь ланцюжок інструментів побудови, тестування та розгортання, ніж при створенні самостійно. [2]

У 2004 році Косуке Кавагучі був Java-розробником у компанії Sun. Кавагучі втомився ламати збірки у своїй розробці і хотів знайти спосіб дізнатись, перед тим, як передавати код у сховище, чи працює код. Тож Кавагучі створив сервер автоматизації на Java, для Java, під назвою Hudson. Hudson став популярним у Sun і поширився на інші компанії, оскільки він був відкритим ПО. У 2011 році, суперечка між Oracle (яка придбала Sun) та незалежною спільнотою відкритих джерел Hudson призвела до вилки зі зміною назви, Дженкінс. У 2014 році Kawaguchi став технічним директором CloudBees, який пропонує продукти постійної доставки на основі Jenkins. Обидві вилки продовжували існувати, хоча Дженкінс був набагато активнішим.

Jenkins поширюється як архів WAR та як пакети інсталятора для основних операційних систем, як пакет Homebrew, як образ Docker та як вихідний код. Вихідним кодом є переважно Java, з кількома файлами Groovy, Ruby та Antlr. Ви можете запустити Jenkins WAR окремо або як сервлет на сервері програм Java,

наприклад Tomcat. У будь-якому випадку він створює веб-інтерфейс користувача та приймає виклики до свого REST API.

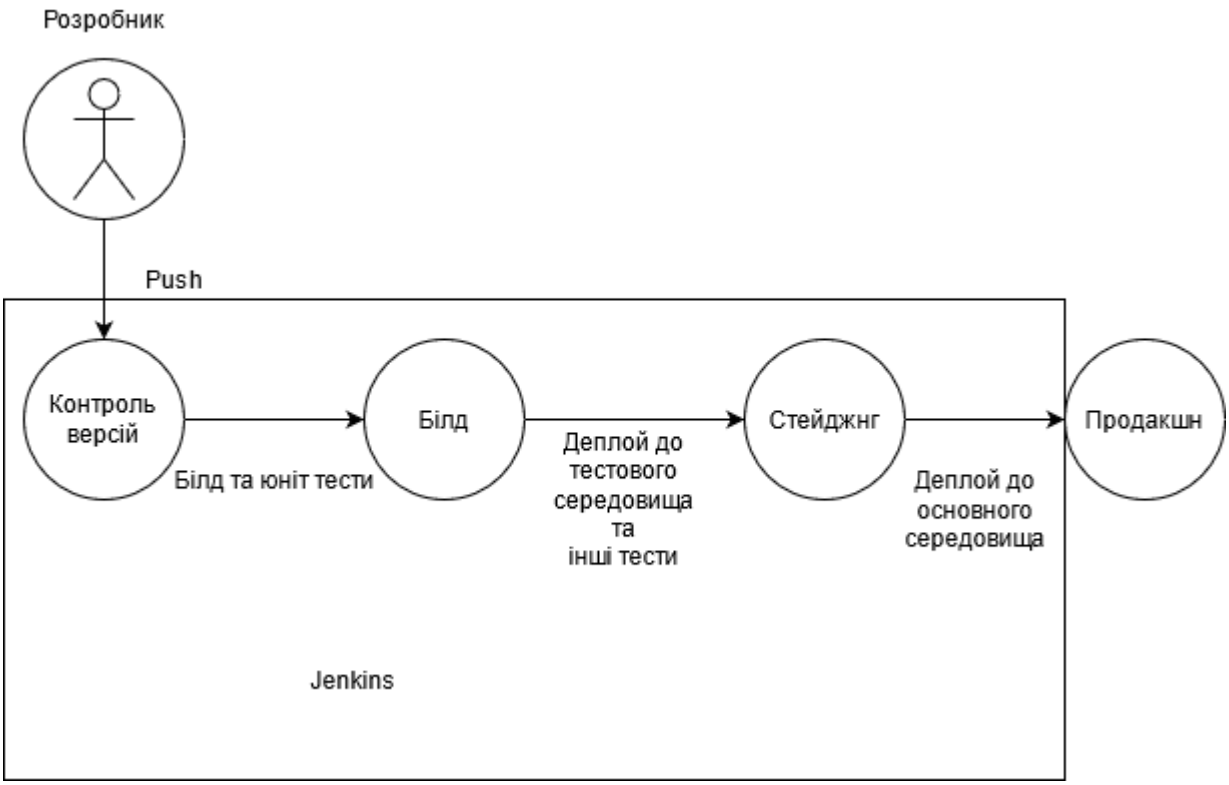
Після встановлення і налаштування Jenkins треба створити проект, які Дженкінс буде збирати. Незважаючи на те, можна використовувати веб-інтерфейс для створення сценаріїв, найкраща практика - створити скрипт pipeline під назвою Jenkinsfile та додати його у свій репозиторій.

5.2. Проектування

Ця курсова робота присвячена проектування і реалізації CI/CD як частину системи яка розроблялась командно з студентами ІПЗ-4 та студентом менеджменту. Мета колаборації - побудова системи пошуку документів по Державному реєстру судових рішень. Оскільки в команді більше ніж 1 людина, існує потреба в зборі оновлень коду від всіх учасників, контролю щоб проект не “ламався” та після перевірки оновлював публічний сайт доступний користувачам, виникла нагода в організації CI/CD.

Була розроблена така схема роботи:

1. Розробник здійснює “push” до системи контролю версії(GitLab) на гілку певного нового функціоналу
2. В Jenkins спрацьовує “webhook”, та відбувається білд та автоматичні тести нової версії системи
3. Після достатньої кількості комітів розробник створює “pull request”
4. Відбуваються “integration” та інші тести
5. Новий функціонал додається до системи



Зображення 6 - Можлива схема CI/CD

5.3. Реалізація

В процесі розробки системи загалом було реалізовано роботу по такій схемі:

1. Розробник завантажує коміт до GitLab
2. Відпрацьовують Unit-тести
3. Відбувається ручне тестування
4. На Heroku завантажується нова версія системи

Зображення 7 - Реалізована схема CI/CD

6. Висновки

В межах даної роботи було підтверджено що процес важливий для організації ефективної роботи над розробкою програмного забезпечення, а саме можливість частіше та надійніше тестувати та інтегрувати в фінальний продукт зміни. Типовий процес складається з таких фаз: планування, кодування, білд, тестування, реліз, деплой, оперування та моніторинг. Для кожної фази існує свій набір технологій та рішень, які мають різні переваги та недоліки.

В результаті роботи було досліджено процес CI/CD, оцінено та обрано одну з систем роботи з CI/CD в процесі проектування, та реалізовано цей процес.

В процесі реалізації були реалізовані фази білду, тестування, релізу, деплою. Завдяки цьому розробка над проектом велася розподіленою командою.

Оскільки робота велась з обмеженим часом, то можна було б розширити фази тестування, моніторингу.

7. Джерела

1. What is CI/CD? Continuous integration and continuous delivery explained [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html
2. What is Jenkins? The CI server explained [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html
3. Top 7 Best CI/CD Tools you should get your hands on in 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/devops-dudes/top-7-best-ci-cd-tools-you-should-get-your-hands-on-in-2020-832c29db936a
4. 9 Reasons Why You Should Use CICD [Електронний ресурс] – Режим доступу до ресурсу: https://thepythonguru.com/9-reasons-why-you-should-use-cicd/