

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

## **Порівняння сучасних рішень для налаштування СІ та CD**

**Текстова частина до курсової роботи  
за спеціальністю „Інженерія програмного забезпечення” 121**

Керівник курсової роботи  
доцент Франчук О.В.  
*(прізвище та ініціали)*

\_\_\_\_\_  
*(підпис)*  
“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

Виконав студент ІПЗ-1  
Гетьман М.С.  
*(прізвище та ініціали)*

“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

Київ 2020

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЇВО-МОГИЛЯНСЬКА АКАДЕМІЯ»  
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Декан факультету інформатики,  
доцент., д.т.н.

\_\_\_\_\_ А. М. Глибовець  
(підпис)  
“       ” \_\_\_\_\_ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Гетьману Максиму

1 курсу магістратури факультету інформатики

ТЕМА: Порівняння сучасних рішень для налаштування CI та CD

Зміст ТЧ до курсової роботи:

1. Вступ
2. Що таке CI/CD?
3. Порівняння готових рішень для CI/CD
4. Висновки
5. Список джерел
6. Додатки (за необхідністю)

Дата видачі “       ” \_\_\_\_\_ 2020 р.

Керівник \_\_\_\_\_ Завдання отримано \_\_\_\_\_

## Календарний план виконання курсової роботи

**Тема:** Порівняння сучасних рішень для налаштування CI та CD

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	Лютий 2020 р.	
2.	Пошук літератури, ознайомлення з теорією про CI/CD	Лютий 2020 р.	
3.	Розгляд існуючих рішень, практична робота з деякими із них	Березень 2020 р.	
4.	Тестування налаштованих програм	Березень 2020 р.	
5.	Створення слайдів для доповіді та написання доповіді	Квітень 2020 р.	
6.	Остаточне оформлення текстової частини та слайдів презентації	Травень 2020 р.	
7.	Захист курсової роботи	Травень 2020 р.	

Студент \_\_\_\_\_ Гетьман М.С. \_\_\_\_\_

Керівник \_\_\_\_\_ Франчук О.В. \_\_\_\_\_

“ \_\_\_\_\_ ” \_\_\_\_\_ р.

# Зміст

<b>Анотація</b>	<b>4</b>
<b>Вступ</b>	<b>5</b>
<b>Розділ 1: Що таке CI/CD?</b>	<b>6</b>
Continuous integration	6
Continuous delivery	9
Continuous deployment	11
Підсумок	12
<b>Розділ 2: Порівняння готових рішень для CI/CD</b>	<b>13</b>
Jenkins	13
TravisCI	19
CircleCI	22
Azure Pipelines	24
<b>Висновки</b>	<b>27</b>
<b>Список використаних джерел</b>	<b>28</b>

## **Анотація**

У даній курсовій роботі було розглянуто, що таке процес CI/CD та які проблеми він вирішує. Було розглянуто деякі рішення для налаштування CI/CD, розглянуто їхні плюси та мінуси, та у яких ситуаціях краще використовувати одне рішення чи інакше.

## Вступ

Continuous integration, continuous delivery чи deployment (CI та CD) - це процеси, що допомагають будувати, тестувати та релізити програмне забезпечення.

Цей підхід набрав велику популярність, і проекти з налаштованими CI/CD процесами дозволяють розробникам швидше писати код, а відповідно команді - швидше випускати новий функціонал.

**Мета** даної курсової роботи - визначити, що таке CI/CD, для чого його використовують, зробити огляд існуючих рішень та порівняти їх.

Роботу поділено на 2 розділи:

У першому розділі розглянуто теоретичні відомості, що таке CI/CD та для чого це використовується.

У другому розділі буде розглянуто та протестовано деякі існуючі рішення для CI/CD.

## Розділ 1: Що таке CI/CD?

Перш ніж перейти до огляду рішень, які вирішують CI/CD, треба насамперед визначити, що означає аббревіатура CI/CD.

### Continuous integration

Одна з великих проблем в розробці програмного забезпечення це інтеграція коду від різних розробників, тобто процес, коли люди починають “зливати” код із різних гілок в одну. Це не є проблемою для поодиноких розробників, але коли багато людей працюють над спільною кодовою базою, при інтеграціях часто трапляються конфлікти чи помилки.

Термін “Continuous integration” вперше був озвучений Граді Бучем ще в 1991, та з часом його тлумачення еволюціонувало. [\[1\]](#)

Ідея CI в тому, щоб зменшити кількість проблем під час таких інтеграцій. Цього можна досягти, якщо інтеграції будуть проводитися часто, тобто кожен розробник буде додавати свої зміни до спільного репозиторію як мінімум раз на день, але чим частіше - тим краще. Таким чином кількість змін в межах однієї інтеграції буде невеликою, і якщо виникає конфлікт, його буде простіше розв’язати.

Патрік Колдвелл у книзі “Code Leader: Using People, Tools, and Processes to Build Successful Software” описує процес Continuous integration як “інтегрувати зміни якомога раніше і частіше”.

Є набір практик та правил, яких слід дотримуватися при імплементації CI[\[2\]](#):

- Використовувати систему контролю версій

- Білди мають бути автоматизовані
- Функціонал має бути покритий тестами
- Коміти до спільного репозиторію робляться мінімум раз на день
- При кожному коміті автоматично створюється білд та запускається набір тестів
- Білди та тести мають проходити швидко (до 10 хвилин)

Важлива ідея в тому, що процес CI має бути автоматизованим. Якщо білд не може бути успішно створеним, або тести не проходять, то розробникам має прийти про це повідомлення, і виправлення помилки має найвищий пріоритет.

Зазвичай процес CI відбувається в такій послідовності:

1. Розробник вносить зміни до коду репозиторія
2. CI-сервер бачить ці зміни (по тригеру, або сам періодично робить перевірку)
3. CI-сервер робить білд проекту, запускає тести, логує всі результати. Якщо сталася помилка - згідно з налаштуваннями відправляється повідомлення розробникам на пошту, в месенджер тощо

Перелік деяких переваг, які надає CI:

- Зменшення ризиків - завдяки частим інтеграціям, які робляться невеликими частинами, простіше знайти місце в коді, яке спричинило помилку
- Автоматизація рутинних процесів - таких як білд, тестування та деплой програмного забезпечення допомагає зберегти час і зменшує ризик людського фактору



- Зазвичай процес CI змушує розробників писати код, який є більш модульним та менш комплексним
- Код покривається тестами
- Завжди доступна “теперішня” версія продукту, яку можна тестувати, показати замовнику чи задеплоїти до продакшину

CI також потребує певної кількості ресурсів, необхідних для його налаштування[3]:

- Тести мають бути написані для всього нового функціоналу та для багфіксів
- Треба налаштувати CI сервер, який буде відповідальним за білд та запуск тестів для проекту при появі нових комітів
- Розробникам треба частіше мерджити свої зміни до основного репозиторію, принаймні раз на день, що не завжди може вдаватися
- На налаштування CI треба додаткові кошти - платити за обчислювальні ресурси, можливо за ліцензію для CI софта, яка може дорого коштувати для великих компаній

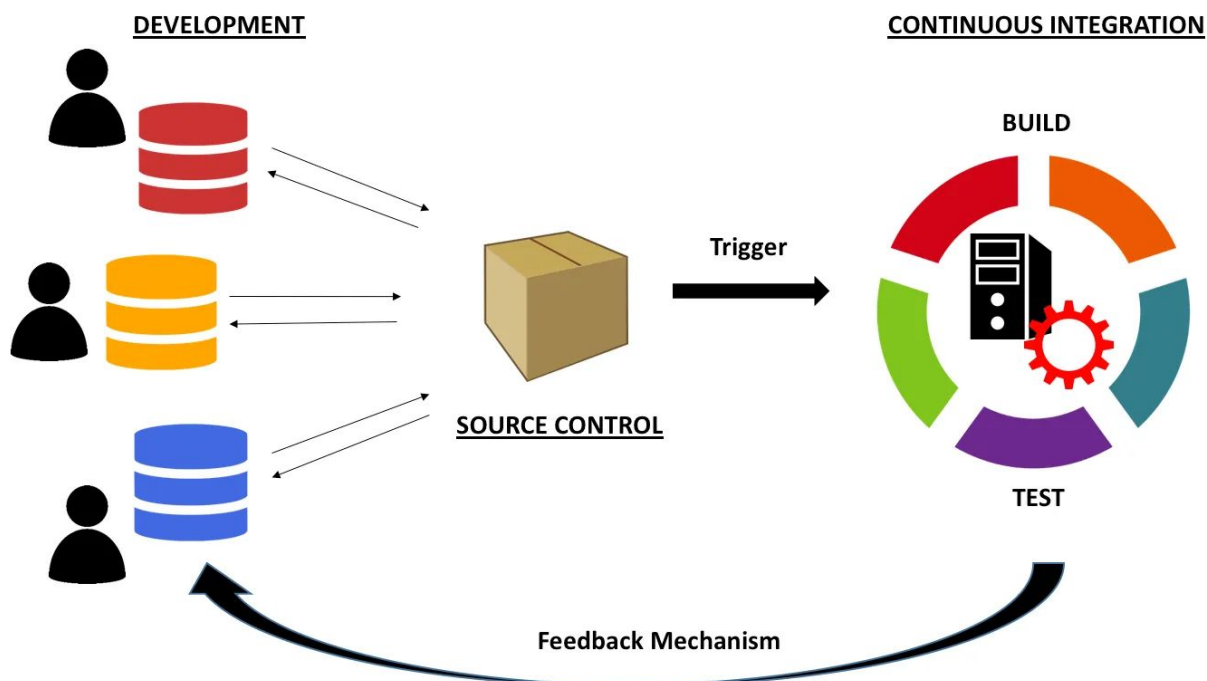


Рис. 1.1

(<http://i1.wp.com/samirbehara.com/wp-content/uploads/2018/04/continuous-integration.png>)

## Continuous delivery

Абревіатура CD означає “Continuous delivery” або “Continuous deployment” (описано в наступному розділі) в залежності від контексту, де вона використовується.

Continuous delivery - це “логічне продовження” CI, яке дозволяє новим змінам швидко опинитися в production середовищі. Таким чином, крім автоматизованого тестування та постійної інтеграції з основним репозиторієм, які надає Continuous integration, процес релізу також автоматизується завдяки Continuous delivery.

У результаті програмне забезпечення можна релізити просто “натиснувши” на кнопку.

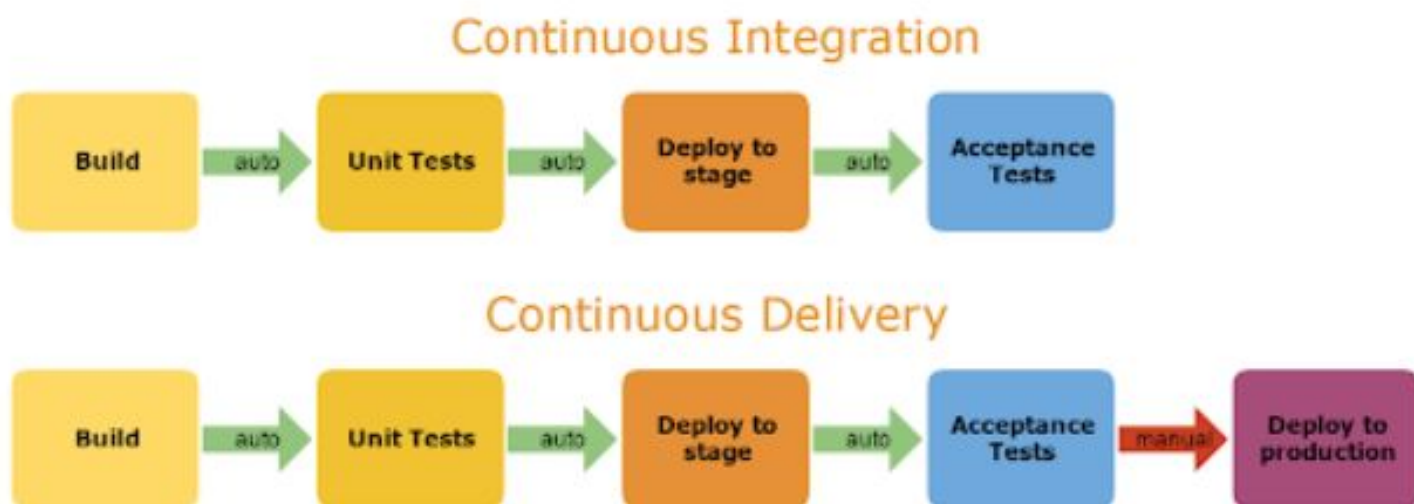


Рис. 1.2

([https://lh3.googleusercontent.com/proxy/PRTUsNm2auhJzPWmQ8DsgjYYW4zIslbrJ5G12UdPfcaJGkcqtdJ7I2kdVDizLXhijy8ms\\_k0UZI9COF85VQGihmHyj2cxiRZTiKcjHGJCpYDQXIxrQevIQVNHAM](https://lh3.googleusercontent.com/proxy/PRTUsNm2auhJzPWmQ8DsgjYYW4zIslbrJ5G12UdPfcaJGkcqtdJ7I2kdVDizLXhijy8ms_k0UZI9COF85VQGihmHyj2cxiRZTiKcjHGJCpYDQXIxrQevIQVNHAM))

Відповідно, завдяки автоматизації, релізи можуть виконуватися частіше, користувачі швидше отримають новий функціонал, а команда розробників збереже свій час.

Переваги, які надає Continuous delivery[4]:

- Релізи відбуваються з меншим ризиком

- Релізи проходять швидше
- Маленькі зміни можна випускати в production, не чекаючи великого релізу

Для успішного налаштування Continuous delivery потрібно виконати зокрема наступні вимоги:

- Налаштований процес Continuous integration
- Виділити ресурси на автоматизацію процесу релізу (ліцензії, сервери, гроші, час розробників)

## Continuous deployment

Continuous deployment (CD) - при його використанні будь-який коміт, який успішно пройшов всі автоматичні тести, потрапляє в production середовище[5]. На відміну від Continuous delivery, немає людини, яка б “запустила” процес релізу. Реліз виконується автоматично після успішного проходження всіх етапів автоматичного тестування, тобто при кожному успішному коміті.

Тому при імплементації Continuous deployment дуже важливо мати якісний набір тестів.

Переваги:

- Користувачі програмного забезпечення бачать постійне покращення продукту, кожного дня, а не раз в місяць, квартал тощо
- Помилку в production середовищі простіше виправити, оскільки зміни додаються невеликими частинами
- Швидка розробка, оскільки не треба робити паузи для запуску релізів

## Підсумок

Автоматизація з використанням CI/CD може збільшити продуктивність команди та зменшити ризики при релізах.

Налаштування CI/CD потребує ресурсів для налаштування, але в результаті отримується багато переваг, описаних в розділі 1.

Схему CI/CD показано на Рис 1.2. У другому розділі буде розглянуто найбільш популярні рішення для CI/CD.

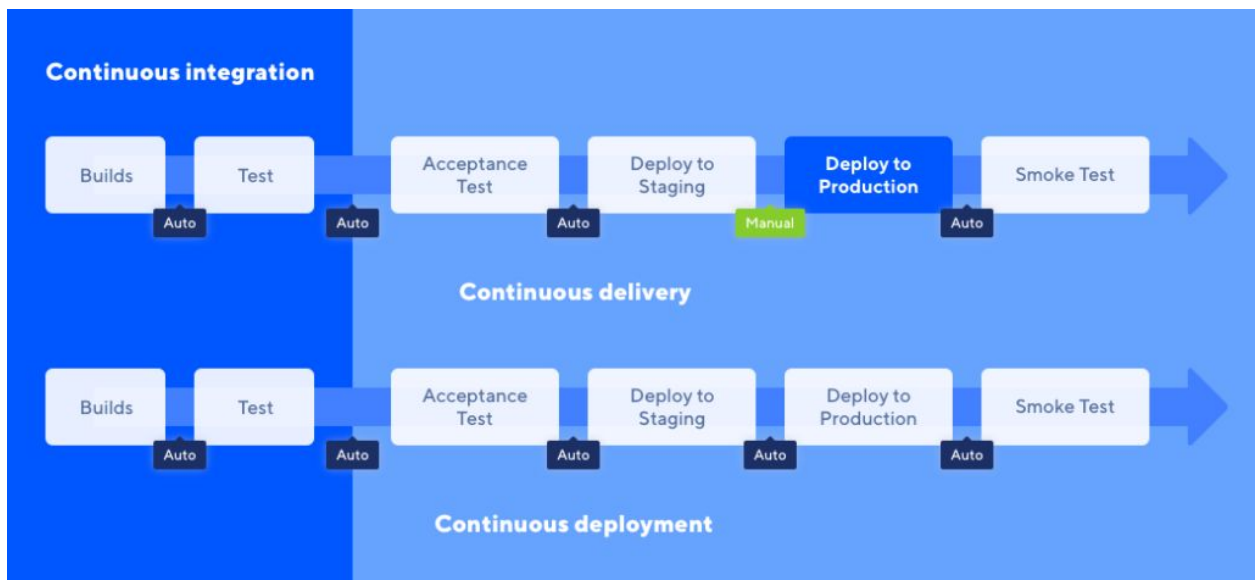


Рис. 1.3

(<https://d1h3p5fzmizjvp.cloudfront.net/wp-content/uploads/2019/09/04164708/CI-CD-CD-1024x473.png>)

## Розділ 2: Порівняння готових рішень для CI/CD

Для тестування CI/CD було створено простий веб сайт на ASP.Net Core 3.1 і ReactJS та розвернуто його в Azure, використовуючи Azure App Service, як зображено на рисунку 2.1. Код зберігався в github репозиторії.

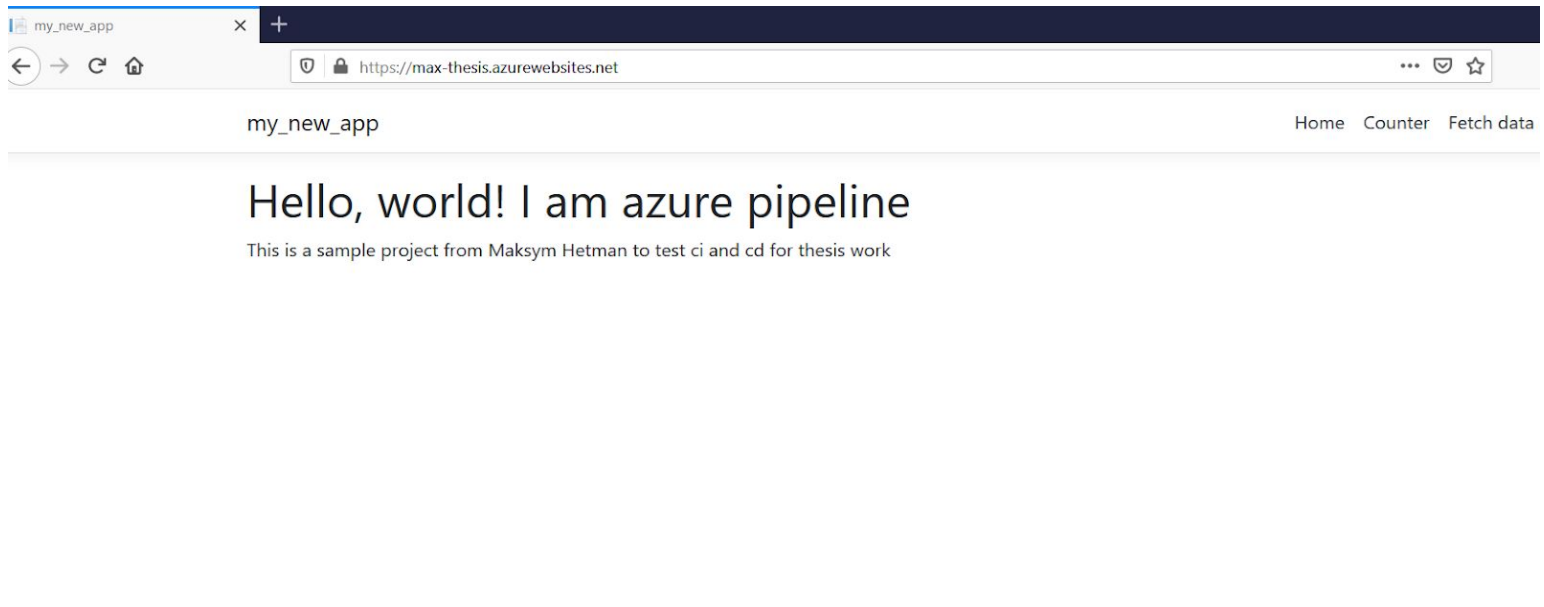


Рис 2.1

(скріншот із <https://max-thesis.azurewebsites.net/>)

Для CI/CD є багато готових рішень, у межах цієї роботи розглянуто деякі із найбільш популярних.

### Jenkins

Jenkins - це найпопулярніше рішення для налаштування CI/CD. Створений у 2004 році, спочатку він мав назву Hudson і в 2011 отримав назву Jenkins після суперечок з Oracle про авторські права.<sup>[6]</sup>



Рис 2.2

(<https://image.slidesharecdn.com/jenkinsci-formacdevops-170211114205/95/jenkins-ci-formacdevops-4-638.jpg?cb=1486813442>)

Основна причина популярності Jenkins - відкритий вихідний код (написаний на Java) і його можливості для розширення. На березень 2020 року Jenkins налічує більше ніж 1500 доступних плагінів. [7]

Дане програмне забезпечення використовується такими компаніями, як Facebook, Netflix, Udemy, Lyft, Twitch, LinkedIn [8]

Jenkins використовує Master-Slave архітектуру і підтримує розподілене налаштування. Є хорошим рішенням для великих проектів.

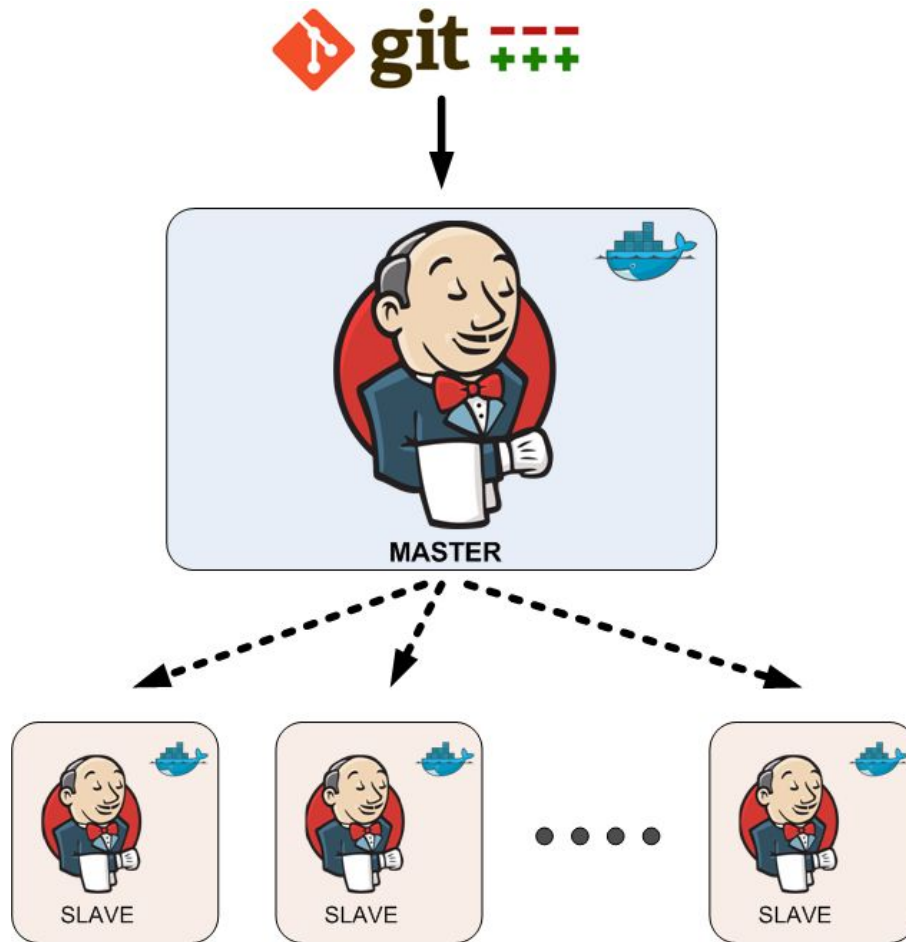


Рис 2.3

([https://miro.medium.com/max/634/1\\*85otVgjdzFKPpKYXpK1JdA.png](https://miro.medium.com/max/634/1*85otVgjdzFKPpKYXpK1JdA.png))

Оскільки написаний на Java, Jenkins є крос-платформенним рішенням та доступний на таких платформах як Windows, macOS, Ubuntu та інші Unix-системи. Також доступний як докер image.

У нашому випадку для швидкого налаштування Jenkins створений через Azure Portal (Рис 2.4). Azure дозволяє заповнити основні поля, необхідні для Jenkins, та сам створює його по кліку кнопки.



Home > Marketplace > Jenkins® > Create Jenkins®

## Create Jenkins®

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Resource group \* ⓘ  [Create new](#)

### Instance details

Region \* ⓘ

Name \*

User name \*  ✓

Authentication type \* Password SSH Public Key

Password \*  ✓

Confirm password \*  ✓

✓ Password and confirmation fields must match.

[Review + create](#)

[< Previous](#)

[Next : Additional Settings >](#)

Рис 2.4

(скріншот з <https://portal.azure.com>)

Після створення Jenkins треба підключитися до нього через SSH чи RDP (в залежності від операційної системи) та поставити програмне забезпечення, яке необхідне, щоб створювати білди для проекту. В нашому випадку це dotnet core SDK та npm, оскільки проект написаний з використанням .net core та reactJs.

Для вирішення більшості задач для Jenkins вже написані плагіни, в тому числі для інтеграції з Azure App Service. Налаштування Jenkins зводиться до того, що треба знайти правильний набір плагінів і правильно налаштувати кожен із них.

Всі налаштування для Jenkins прописуються в пайплайнах. Пайплайни можна створювати і зберігати як в самому Jenkins, так і окремо в репозиторіях (файл має називатися Jenkinsfile), щоб можна було відслідковувати історію змін та перевіряти налаштування пайплайну, використовуючи мердж реквести.

Приклад простого пайплайну показано на рисунку 2.4. У ньому також використанні змінні середовища, зокрема лінк до репозиторія на гітхабі та назва ресурсу групи і назва апікейшна в azure app service.

```
1 node {  
2   stage('Fetch code') {  
3     git branch: 'master', url: params.git_repo  
4   }  
5  
6   stage('Publish') {  
7     sh 'dotnet publish -c Release '  
8   }  
9  
10  stage('Deploy to app service') {  
11    azureWebAppPublish azureCredentialsId: params.azure_cred_id,  
12    resourceGroup: params.res_group, appName: params.app_name, sourceDirectory: "bin/Release/netcoreapp3.1/publish/"  
13  }  
14 }
```

Рис 2.5

*(скріншот з sublime text)*

Як виглядає інтерфейс Jenkins можна побачити на рис 2.6

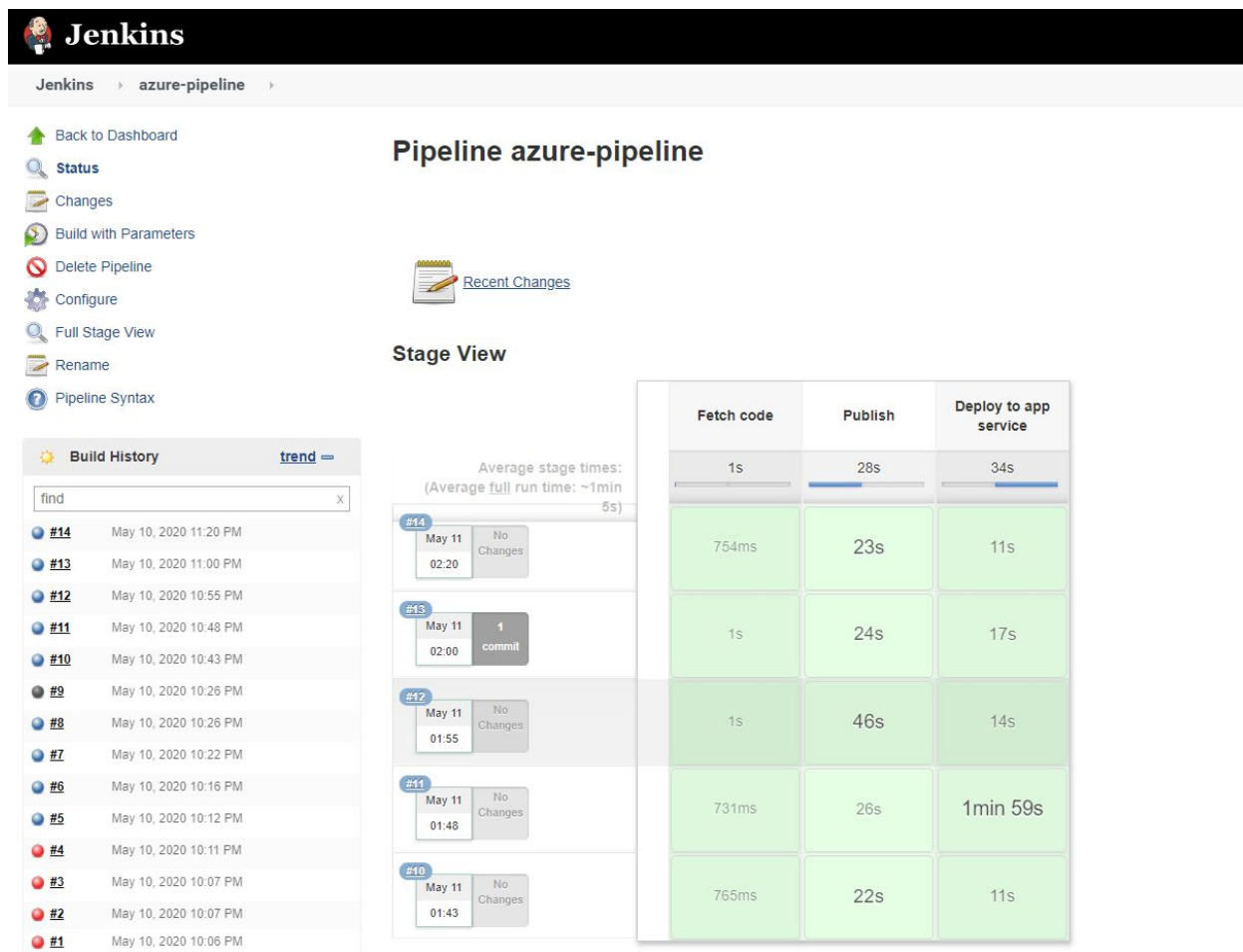


Рис 2.5

(скріншот з кабінету в Jenkins)

Плюси Jenkins:

- Відкритий вихідний код
- Безкоштовний
- Велика кількість плагінів
- Крос-платформенність
- Популярність (велике ком'юніті, багато інформації)

Мінуси:

- Документація
- Налаштування може зайняти більше часу, ніж в конкурентів

## TravisCI

TravisCI - написане на Ruby програмне забезпечення для CI/CD. Легко інтегрується з системою контролю версій github.



Рис 2.6

([https://miro.medium.com/max/600/1\\*VXdK53mBfr27iT8LiHNAbg.png](https://miro.medium.com/max/600/1*VXdK53mBfr27iT8LiHNAbg.png))

TravisCI використовують такі компанії, як Lyft, Heroku, DuckDuckGo, Sentry та інші[9].

Перші 100 білдів у TravisCI безкоштовні, далі пропонується вибрати із різних варіантів підписок, як показано на рисунку 2.7.

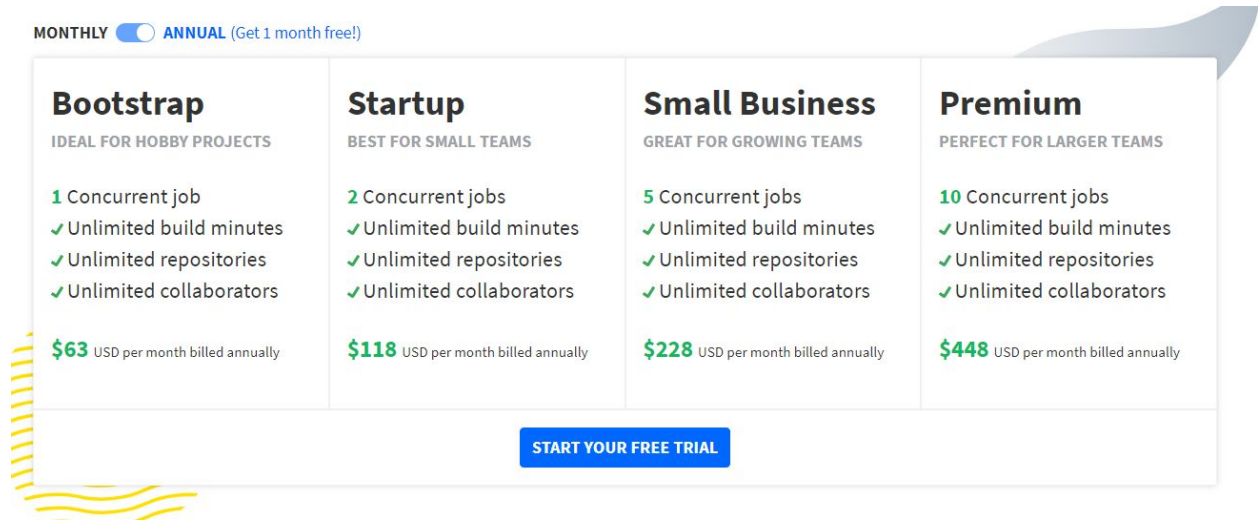


Рис 2.7

(скріншот із <https://travis-ci.com/plans>)

Всі рішення працюють в хмарі та відрізняються тільки кількістю паралельних запусків. Існує також Enterprise версія, яку можна поставити на on-premise сервери.

TravisCI є безкоштовним для проектів із відкритим вихідним кодом ізі надає для них безкоштовно 5 паралельних запусків та нелімітовану кількість білдів.

Всі налаштування для CI/CD зберігаються в YAML форматі в корні репозиторію з кодом.

Приклад такого файлу показано на рисунку 2.8

```
1  language: csharp
2  mono: none
3  dotnet: 3.1.200
4  install:
5  - dotnet restore
6  script:
7  - dotnet publish -c Release
8  deploy:
9  provider: azure_web_apps
10 username: $AZURE_WA_USERNAME
11 password: $AZURE_WA_PASSWORD
12 site: $AZURE_WA_SITE
```

Рис 2.8

*(скріншот із sublime text)*

Плюси TravisCI:

- Легко налаштовується
- Доступний в клауді
- Зрозумілий UI
- Безкоштовний для open source
- Зрозуміла документація

Мінуси:

- Розуміє репозиторії тільки github і bitbucket(beta версія)
- Немає безкоштовного плану (тільки для open source проектів)

## CircleCI

CircleCI - рішення для CI/CD, яке використовується такими компаніями, як Stripe, StackShare, Fiverr, 9gag та іншими.



Рис 2.9

*(<https://d3r49iyjzglexf.cloudfront.net/circleci-logo-stacked-fb-657e221fda1646a7e652c09c9fbfb2b0feb5d710089bb4d8e8c759d37a832694.png>)*

Компанія створена в 2011 році. CircleCI працює в хмарі, але також надає рішення для on-premises налаштування. Є безкоштовна версія, в якій є певна кількість кредитів для використання білд машин, також в безкоштовній версії відсутня підтримка білдів для macOS.

Варіанти з різними цінами можна подивитися на рис 2.9 Також CircleCI підтримує розробку з відкритим вихідним кодом і має безкоштовну пропозицію для таких проектів.

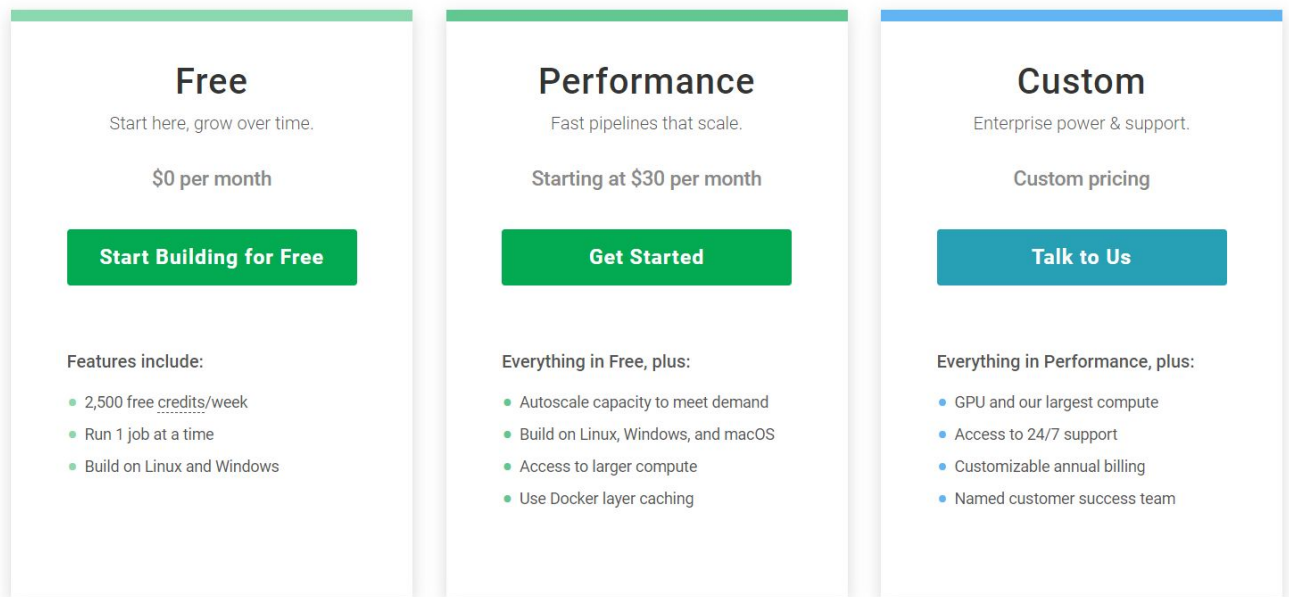


Рис 2.10

(скріншот із <https://circleci.com/pricing/>)

Налаштування для CircleCI робиться в YAML форматі, файли мають знаходитися в репозиторії з кодом

Плюси CircleCI:

- Легко налаштовується
- Швидка служба підтримки, яка відповідає не більше ніж через 12 годин
- Доступний в клауді
- Є безкоштовна версія
- Є безкоштовна пропозиція для проектів з відкритим вихідним кодом



Мінуси:

- Не відразу зрозумілий UI

## Azure Pipelines

Azure Pipelines - CI/CD рішення від компанії Microsoft, яке є частиною SaaS продукту “Azure DevOps”.

Azure DevOps - це хмарне рішення, але також є on-premise версія. Це платформа, на якій можна створювати гіт-репозиторії, борди для задач, тест плани тощо. І одним із доступного функціоналу є CI/CD, який має назву Azure Pipelines.



Рис 2.11

([https://miro.medium.com/proxy/1\\*fs244dG4ZD3OwinIf17Kew.jpeg](https://miro.medium.com/proxy/1*fs244dG4ZD3OwinIf17Kew.jpeg))

Це рішення використовується такими компаніями як Microsoft, accuRx, OutSystems та іншими.

Є безкоштовний варіант Azure Pipelines, який надає 1800 хвилин CI/CD на ресурсах Microsoft та 1 self-hosted job з безлімітною кількістю хвилин на self-hosted ресурсах. Azure Pipelines можна використовувати або як частину Azure DevOps, або як індивідуальний сервіс. Ціни можна побачити на рисунку 2.12





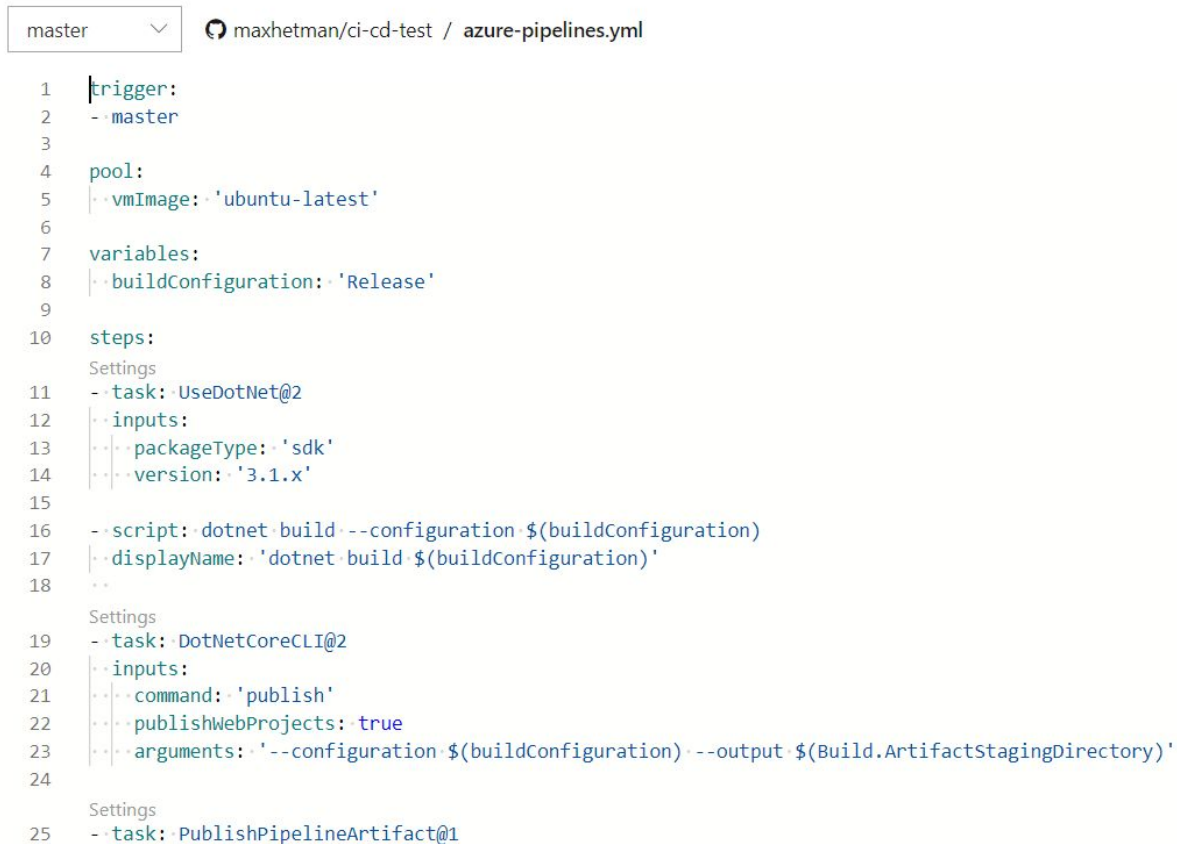
Azure Pipelines	Azure Artifacts	Basic Plan	Basic + Test Plans
 <p>1 Free Microsoft-hosted CI/CD 1 Free Self-Hosted CI/CD</p> <p><b>Start free &gt;</b></p> <ul style="list-style-type: none"> <li>1 Microsoft-hosted job with 1,800 minutes per month for CI/CD and 1 self-hosted job with unlimited minutes per month</li> <li>\$40 per extra Microsoft-hosted CI/CD parallel job and \$15 per extra self-hosted CI/CD parallel job with unlimited minutes</li> </ul>	 <p>2 GB free, then starting at \$2 per GB</p> <p><b>Start free &gt;</b></p> <ul style="list-style-type: none"> <li>Industry-leading NuGet Server</li> <li>Support for Maven, npm, and Python packages</li> <li>Upstream sources to help protect open-source dependancies</li> <li>Integrated with Azure Pipelines</li> <li>Sophisticated access controls</li> </ul>	 <p>First 5 users free, then \$6 per user per month</p> <p><b>Start free &gt;</b></p> <ul style="list-style-type: none"> <li><b>Azure Pipelines:</b> Includes the free offer from INDIVIDUAL SERVICES</li> <li><b>Azure Boards:</b> Work item tracking and Kanban boards</li> <li><b>Azure Repos:</b> Unlimited private Git repos</li> <li><b>Azure Artifacts:</b> 2 GB free per organization</li> <li>Load testing (20,000 VUMs/month)</li> </ul>	 <p>\$52 per user per month</p> <p><b>30 day free trial &gt;</b></p> <ul style="list-style-type: none"> <li>Includes all Basic plan features</li> <li>Test planning, tracking &amp; execution</li> <li>Browser-based tests with annotation</li> <li>Rich-client test execution</li> <li>User acceptance testing</li> <li>Centralized reporting</li> </ul>

Рис 2.12

(скріншот із <https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/>)

Має чудову інтеграцію з Azure, зрозумілий інтерфейс, конфігурація зберігається в YAML форматі.

На рис 2.13 показано приклад YAML конфігурації



```

1 trigger:
2   - master
3
4 pool:
5   - vmImage: 'ubuntu-latest'
6
7 variables:
8   - buildConfiguration: 'Release'
9
10 steps:
11   Settings
12   - task: UseDotNet@2
13     inputs:
14       - packageType: 'sdk'
15       - version: '3.1.x'
16   - script: dotnet build --configuration $(buildConfiguration)
17     displayName: 'dotnet build $(buildConfiguration)'
18   -
19   Settings
20   - task: DotNetCoreCLI@2
21     inputs:
22       - command: 'publish'
23       - publishWebProjects: true
24       - arguments: '--configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)'
25   Settings
26   - task: PublishPipelineArtifact@1

```

Рис 2.13

(скріншот із <https://dev.azure.com/>)

Azure Pipelines підтримує будь яку мову програмування, всі популярні системи контролю версій та з його використанням можна деплоїти в Azure, AWS, GCP чи on-premises сервери.

Плюси:

- Зрозумілий UI
- Інтеграція з Azure
- Доступна безкоштовна версія
- Клауд рішення

Мінуси:

- Інтеграція з продуктами не від Microsoft потребує зусиль

## Висновки

Було визначено для чого використовується підхід CI/CD, які проблеми він вирішує, які має переваги та недоліки.

Також було розглянуто, яке програмне забезпечення можна використати, щоб імплементувати підхід CI/CD.

Із існуючих рішень було розглянуто Jenkins, TravisCI, CircleCI та Azure Pipelines.

Було знайдено їхні сильні та слабкі сторони, і в залежності від ситуації можна обрати різний варіант.

Часто немає відповіді на питання, яке саме рішення буде найкращим в конкретній ситуації. Все залежить від багатьох обставин - бюджет, досвід, існуюча інфраструктура тощо.

Наприклад, якщо бюджет невеликий - можливим варіантом є Jenkins, оскільки він безкоштовний. Якщо в компанії вже використовуються сервіси від Azure, можна дивитися в сторону Azure Pipelines, через хорошу інтеграцію. Можливо, хтось із розробників має великий досвід роботи із CircleCI чи Travis і це допоможе команді зекономити багато часу на налаштуваннях.

Крім розглянутих чотирьох варіантів, на ринку є багато інших, такі як GitlabCI, Bamboo, Codeship та інші.

## Список використаних джерел

1. <https://dzone.com/articles/continuous-integration-and-its-whereabouts>
2. [https://en.wikipedia.org/wiki/Continuous\\_integration](https://en.wikipedia.org/wiki/Continuous_integration)
3. <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>
4. <https://continuousdelivery.com/>
5. <https://rollout.io/blog/whats-the-difference-between-continuous-delivery-vs-continuous-deployment/>
6. <https://uk.wikipedia.org/wiki/Jenkins>
7. <https://plugins.jenkins.io/>
8. <https://stackshare.io/jenkins>
9. <https://stackshare.io/travis-ci>