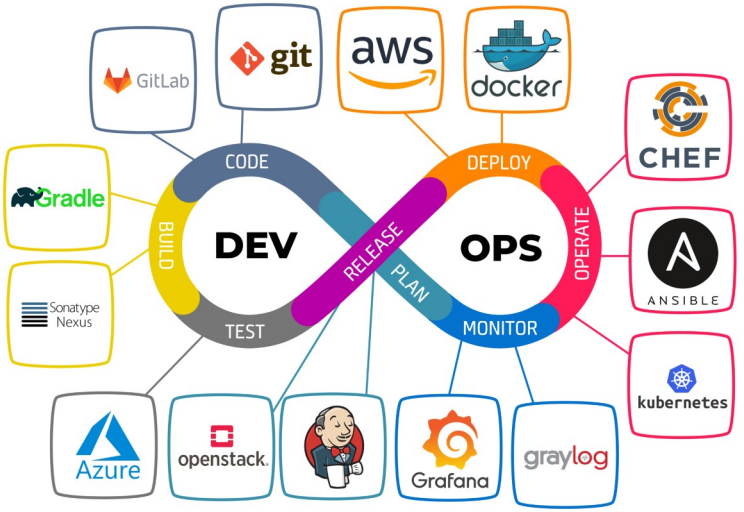


Вступ до DevOps

Лекція 1. Вступ до DevOps

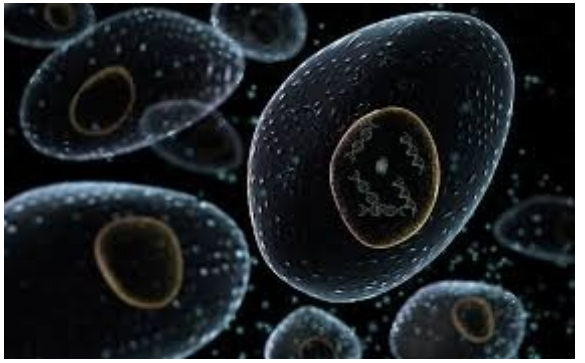
План

1. Історичний аспект.
2. Хмарні технології.
3. DevOps.
4. Контейнери.
5. Kubernetes.
6. Дорожня мапа DevOps.

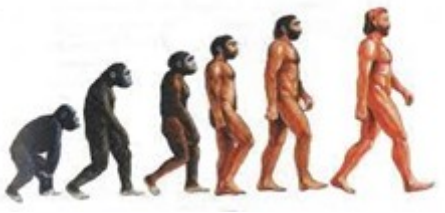
1. Історичний аспект

Великий вибух –
13,7 млрд років тому

Сонячна система –
4,6 млрд років тому



Поява життя на Землі –
3,7 млрд років тому



Поява людини –
6 млн років тому

Когнітивна революція –
70 тис років тому

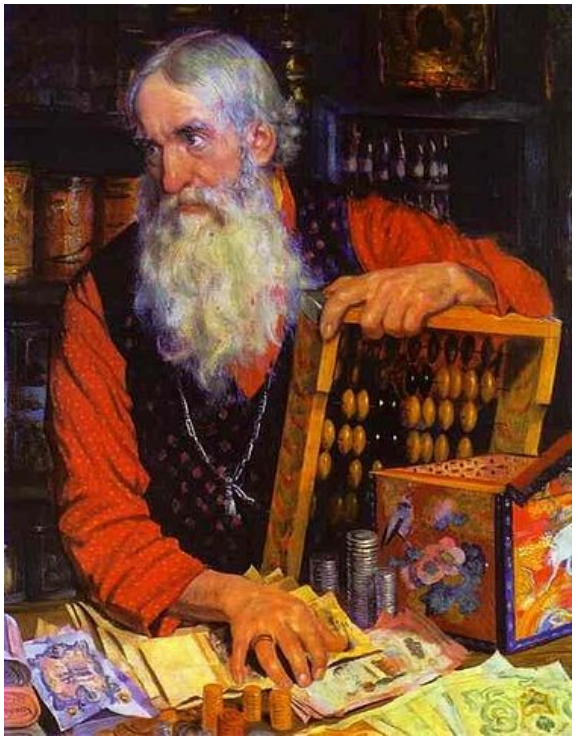
Аграрна революція –
12 тис років тому

Наукова революція –
500 років тому

1. Історичний аспект

Абак (лічильна дошка)
З’явився приблизно у третьому тисячоріччі до н.е. у Стародавньому Вавилоні

Рахівниця
З’явилася як розвиток китайського рахункового пристрою суаньпань приблизно у XIV столітті і активно використовуються досі!!!



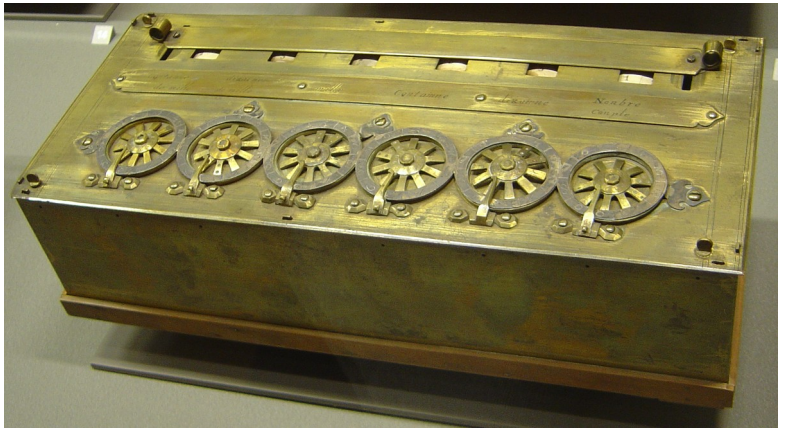
«Купець»
Б. М. Кустодієв, 1918 р.

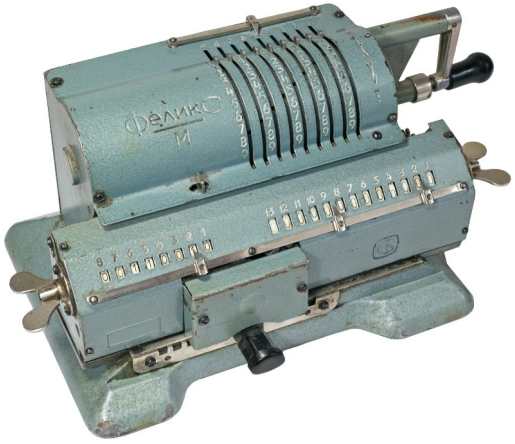
1. Історичний аспект

Блез Паскаль
(1623 – 1662)

Паскаліна (калькулятор Паскаля)

Один з перших у світі механічних обчислювальних пристроїв, створений французьким вченим Блезом Паскалем у 1642 році.

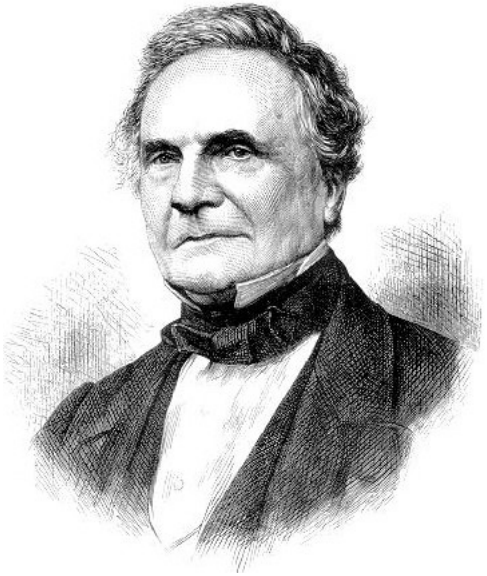




Арифмометр

Портативна механічна обчислювальна машина, призначена для точного множення та ділення. Випускалася у СРСР до 1978 року.

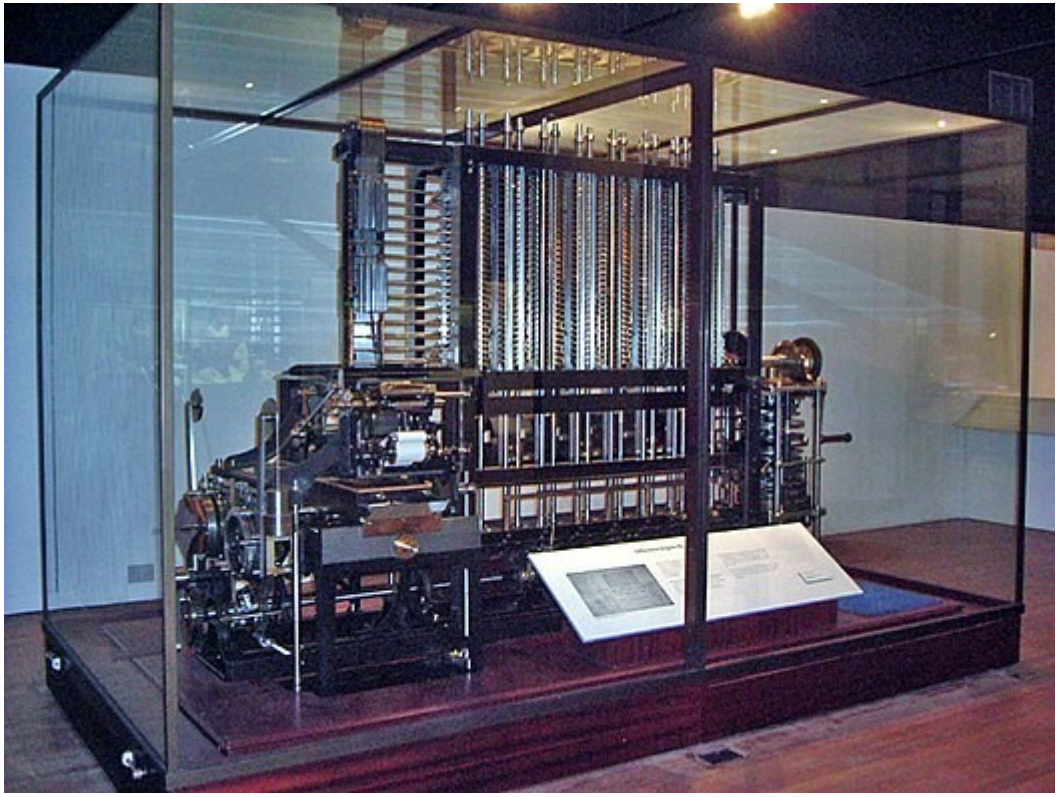
1. Історичний аспект



Чарльз Беббідж
(1791 – 1871)

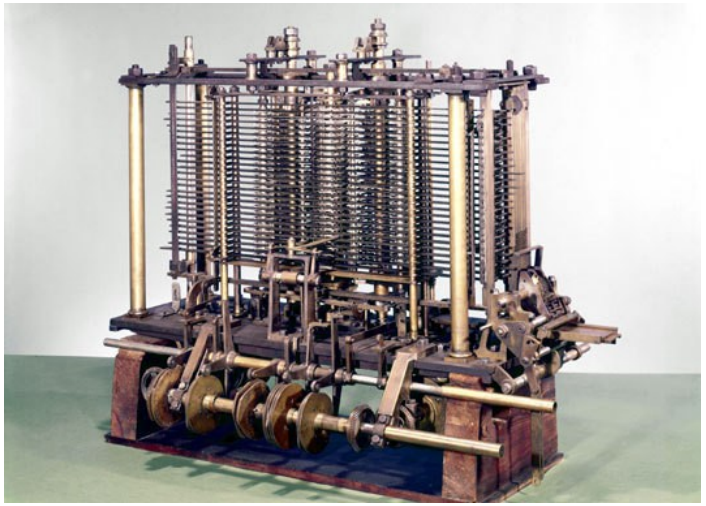
Різницева машина Беббіджа

Механічний пристрій, призначений для автоматичного обчислення значення багаточленів до шостого степеня з точністю до 18-го знака. Був запропонований англійським вченим Чарльзом Беббіджем у 1822 році.



Сучасна реконструкція
різницевої машини Беббіджа

1. Історичний аспект



Аналітична машина Беббіджа

Прообраз сучасного комп’ютера, який складався з арифметичного пристрою («млина»), пам'яті («складу») та пристрою введення-виводу, реалізованого за допомогою перфокарт, що могли бути використані як для введення даних в машину, так і для збереження результатів обчислень, якщо пам’яті було замало.

Ткацький верстат з картами Жаккара

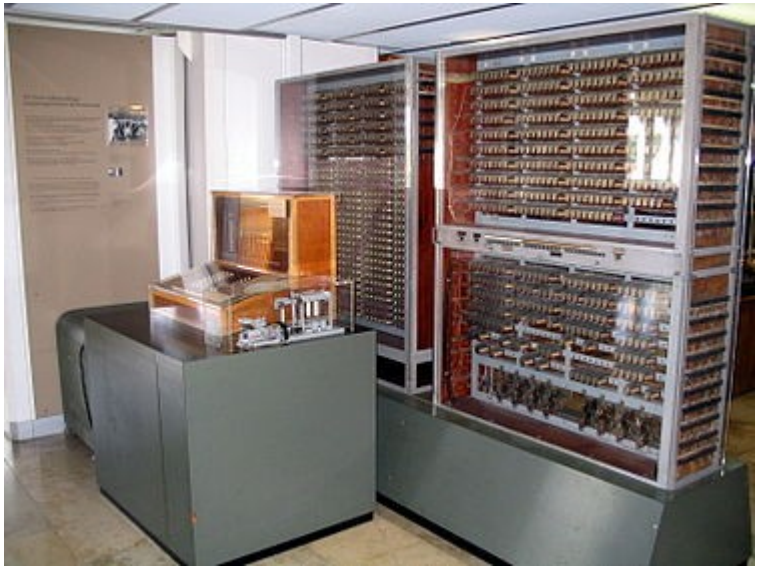
Ада Лавлейс (1815 – 1852)

Ада Лавлейс вважається першим програмістом, оскільки вона вперше описала алгоритм обчислення чисел Бернуллі на аналітичній машині

1. Історичний аспект

Конрад Цузе
(1910 – 1995)

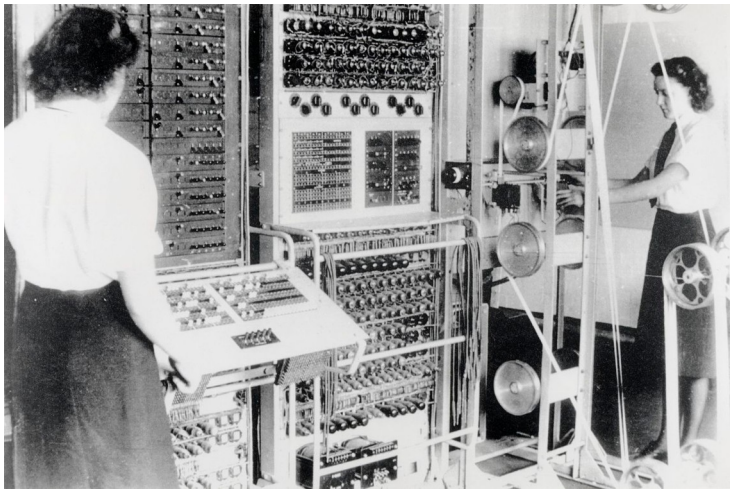
Z3 – перша працездатна повнофункціональна програмно керована обчислювальна машина (фактично – перший електромеханічний комп'ютер). Побудована у 1941 році в Німеччині інженером Конрадом Цузе.



Тактова частота: 5,3 Гц.

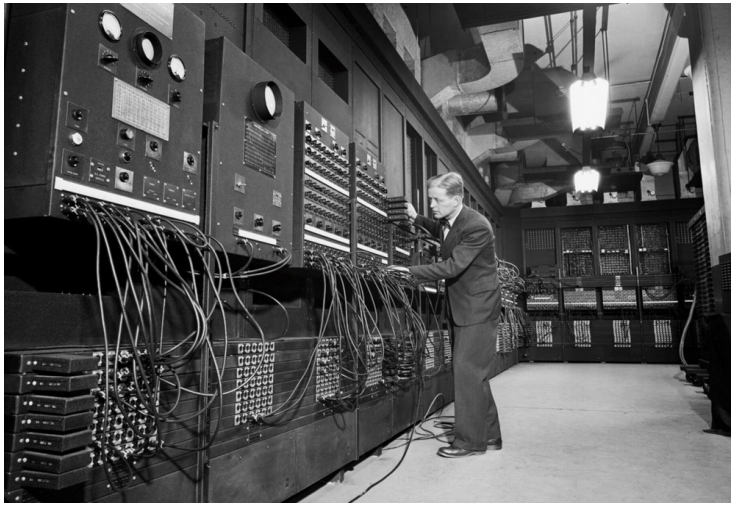
Цузе розробив першу у світі високорівневу мову програмування **Планкалкюль** (Plankalkül – обчислення планів).

1. Історичний аспект



Colossus

Перший в світі повністю електронний секретний британський комп’ютер, побудований у 1943. Використовувався для розшифрування перехоплених німецьких радіоповідомлень. Складався з 1500 електронний ламп. Використання Colossus дозволило скоротити час розшифрування перехоплених повідомлень з кількох тижнів до кількох годин.



ENIAC (Electronic Numerical Integrator and Computer)

Перший електронний цифровий обчислювач загального призначення, який можна було перепрограмувати на вирішення широкого спектра задач. Побудований у 1945 році у США. Мав вагу – 30 тон та тактова частоту – 100 кГц.

1. Історичний аспект

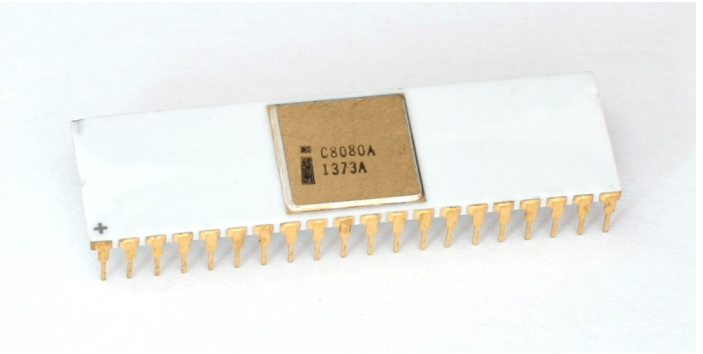
LEO
Перший приватний комп’ютер, який використовувався для обробки комерційних даних (використовувався для розрахунку ціни на продукти харчування). Побудований у 1951 році в Англії). Мав тактову частоту 500 КГц.

UNIVAC (UNIVersal Automatic Computer) – перший комерційний комп’ютер, створений в Сполучених Штатах (1951 рік). Важив 13 тон і мав тактову частоту 2,25 МГц.

1. Історичний аспект

Altair 8800

Перший у світі персональний комп'ютер (мікрокомп'ютер). Розроблений у США у 1975 р. Мав тактову частоту – 2 МГц.

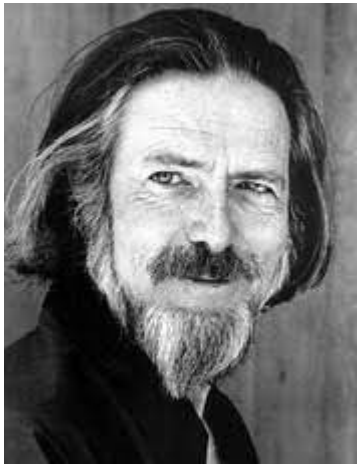


8-бітний мікропроцесор Intel 8080 (1974).

2. Хмарні технології

У світу ніколи не було моменту початку, оскільки він йде по колу, а у кола немає такого місця, де б воно починалося.

Алан Уотс



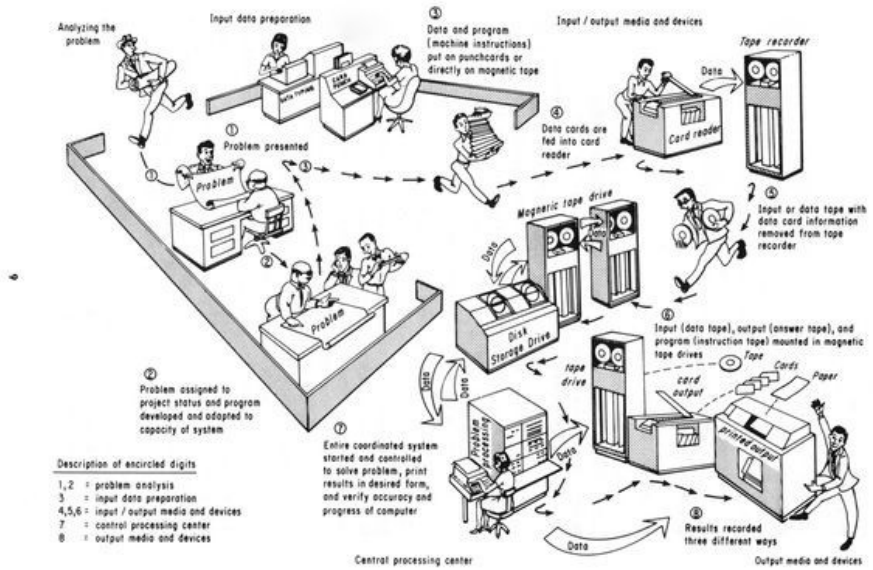
Алан Уілсон Уотс
(1915 – 1973)
британський філософ і письменник

2. Хмарні технології

Комп’ютери першого-третього покоління (мейнфрейми) (1945-1975) були дуже дорогими і складними в обслуговуванні.

Більшість користувачів з ними напряму не контактували. Вони відправляли операторам завдання на виконання і очікували отримання результатів.

Таким чином, користувачі фактично поділяли між собою обчислювальну інфраструктуру і оплачували лише витрачений процесорний час.



Більшості компаній та організацій було невигідно купувати та обслуговувати власні комп’ютери, що призвело до формування наступної бізнес-моделі: **користувачі ділили обчислювальні потужності віддалених комп’ютерів, що належали третій стороні.**

Отже, розподіл доступу до обчислювальних ресурсів з оплатою за використання, є старою ідеєю, яка в наш час називається **«хмарою»**.

2. Хмарні технології

Основна ідея хмарної технології наступна: замість придбання потужного комп’ютера користувач купує його обчислювальний час.

Це дає наступні переваги: замість великих капіталовкладень у комп’ютерне обладнання (яке часто ламається, швидко застаріває та важко масштабується) компанія просто купує час на комп’ютері, що належить третій стороні, і, таким чином, позбавляє себе проблем, пов’язаних з масштабуванням, обслуговуванням та оновленням.

2. Хмарні технології

Використання хмарних технологій дозволяє організаціям **замінити капітальні витрати на операційні**.

Капітальні витрати (CAPEX – capital expenditure) – капітал, що використовується компаніями для придбання або модернізації фізичних активів (житлової та промислової нерухомості, обладнання, технологій тощо).

Операційні витрати (OPEX – operating expense) – повсякденні витрати компанії для ведення бізнесу, виробництва продуктів та послуг.

Хмарний сервіс – це не просто віддалені обчислювальні ресурси, які орендуються, а й розподілені системи. Компанія можете купити безпосередньо обчислювальний ресурс та використовувати його для запуску власного програмного забезпечення, але все частіше в наш час орендується не лише чуже «залізо», а й «софт». Це називається **програмне забезпечення як послуга** (SaaS – software as a service).

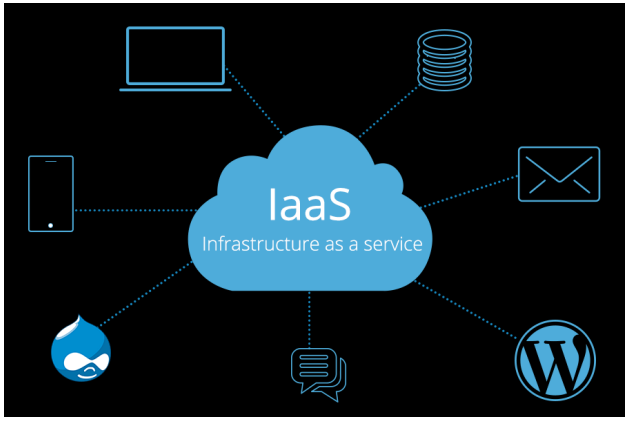
2. Хмарні технології

В разі, якщо компанія використовує хмарну інфраструктуру для запуску власних сервісів, вона купує **інфраструктуру як послугу** (IaaS – infrastructure as a service).

IaaS не потребує капіталовкладень, її не потрібно налагоджувати або оновлювати (це такий самий товар, як вода або опалення).

Організація може брати в оренду програмне забезпечення, яке вона не розробляє власноруч (операційні системи, системи управління базами даних і т. п.).

Компанії, які надають хмарні сервіси (Amazon, Google, Microsoft, ...), також забезпечують все, що необхідне для ефективного виконання програм: кластеризацію, реплікацію, моніторинг, підтримку мережі, обробку черг та потоків і таке інше.



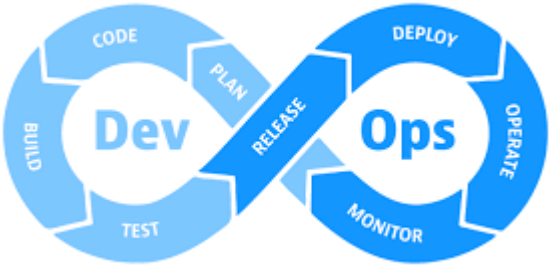
3. DevOps

Історично розробка програмного забезпечення та його системне адміністрування були різними напрямами діяльності IT-бізнесу.

Розробники (developers) – писали програмний код і передавали його для впровадження та обслуговування **адміністраторам** (operations).

Зазвичай розробники й адміністратори рідко перетиналися, оскільки їх цілі та задачі різні: програміст фокусується на швидкій реалізації нових можливостей програм, а системний адміністратор опікується надійністю та стабільністю роботи програмного забезпечення у довгостроковій перспективі.

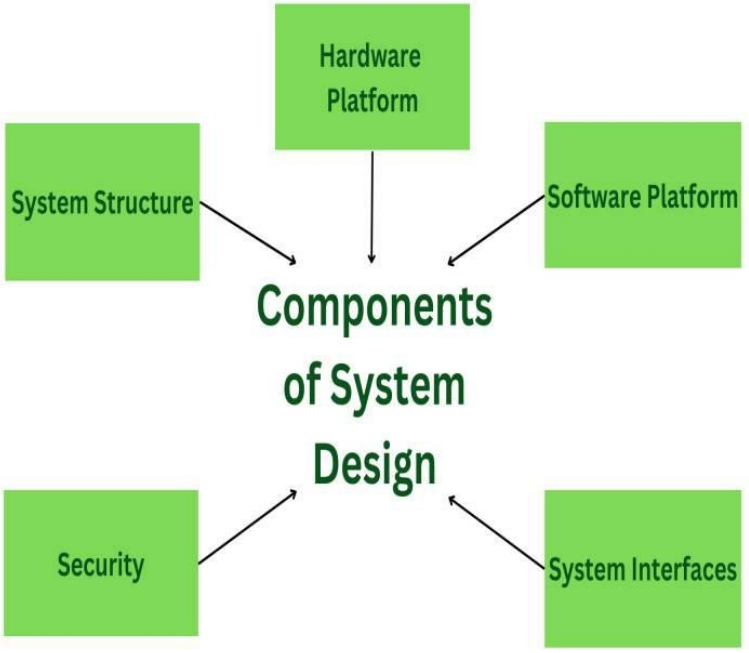
З появою хмарних технологій ця ситуація стала змінюватися: організація роботи програм у хмарі стала більш складною, що стало вимагати від розробників розуміння, як програмне забезпечення буде взаємодіяти з іншими компонентами інфраструктури, а від адміністраторів – як програма працює (або чому вона не працює).



3. DevOps

Сучасна програмна система – це не лише її початковий код, а й відповідне «залізо», цільова операційна система, допоміжне програмне забезпечення, хмарні сервіси, комп’ютерні мережі, засоби безпеки (брандмауери та антивірусні програми), інструменти моніторингу тощо.

Всі ці компоненти тісно пов’язані один з одним, що вимагає від програміста розуміння, як це програмне забезпечення працює.



3. DevOps

DevOps (development & operations) – це сучасна методологія автоматизації технологічних процесів розробки, налаштування та розгортання програмного забезпечення, яка передбачає активну взаємодію фахівців з розробки (developers) з фахівцями з інформаційно-технологічного обслуговування (operations). Це дає можливість досягнення взаємної інтеграції їх технологічних процесів один з одним для забезпечення високої якості програмного продукту.

3. DevOps

Початково DevOps створювався як спроба звести дві групи фахівців воєдино: для співпраці, налагодження взаєморозуміння, поділу відповідальності за надійність системи та коректність програмного коду, а також для покращення масштабованості.

Не зважаючи на те, що методологія DevOps на сьогодні цілком сформувалася, остаточного розуміння, що ж таке DevOps все ще немає.

Так, наприклад, **Джон Віллiс** (John Willis), відомий автор книг про DevOps, визначає чотири ключові аспекти, на яких заснована дана концепція: 1) **культура**, 2) **автоматизація**, 3) **вимірювання** та 4) **обмін знаннями** (CAMS – culture, automation, measurement, sharing).

Брайан Доусон (Brian Dawson) з компанії Cloudbees пропонує інше визначення, яке він називає трійцею DevOps: 1) **люди та культура**, 2) **процес і практика**, 3) **інструменти та технології**.

3. DevOps

Є й альтернативна точка зору: завдяки хмарним технологіям та контейнерам DevOps вже більше не потрібний. Ці погляди називають **NoOps**. Вони базуються на ідеї, що все системне адміністрування зараз делегується хмарному провайдеру або іншому сторонньому сервісу, а отже компаніям не потрібні штатні системні адміністратори.

Помилковість цієї думки полягає в неправильному розумінні того, що на насправді входить у роботу DevOps.

Завдяки DevOps більшість традиційного системного адміністрування виконується до того, як код досягає промислового середовища. Випуск кожної версії передбачає моніторинг, ведення журналу та A/B-тестування. Конвеєри CI/CD автоматично виконують модульні тести, сканування безпеки та перевірку політик при кожній фіксації змін. Розгортання відбувається автоматично. Контроль, завдання та нефункціональні вимоги тепер опрацьовуються перед випуском версії, а не під час або після критичного збою у роботі.

Джордан Бах (AppDynamics)

3. DevOps

З точки зору бізнесу DevOps має таке визначення: «**Поліпшення якості програмного забезпечення шляхом прискорення циклу випуску версій за допомогою хмарних методик та автоматизації з додатковою перевагою у вигляді того, що програмне забезпечення насправді продовжує працювати у промислових умовах**» (The Register).

Впровадження DevOps вимагає від компанії глибокого культурного перетворення, яке має починатися на керівному, стратегічному рівні та поступово поширюватися на кожен відділ організації. Швидкість, динамічність, взаємодія, автоматизація та якість програмного забезпечення є ключовими цілями DevOps, що для багатьох компаній означає суттєві зміни в способі мислення.

DevOps – це не вигадка, а, швидше, спосіб, за допомогою якого успішні організації в наші дні індустріалізують доставку якісного програмного забезпечення; вже завтра та на найближчі роки це стане новою нормою.

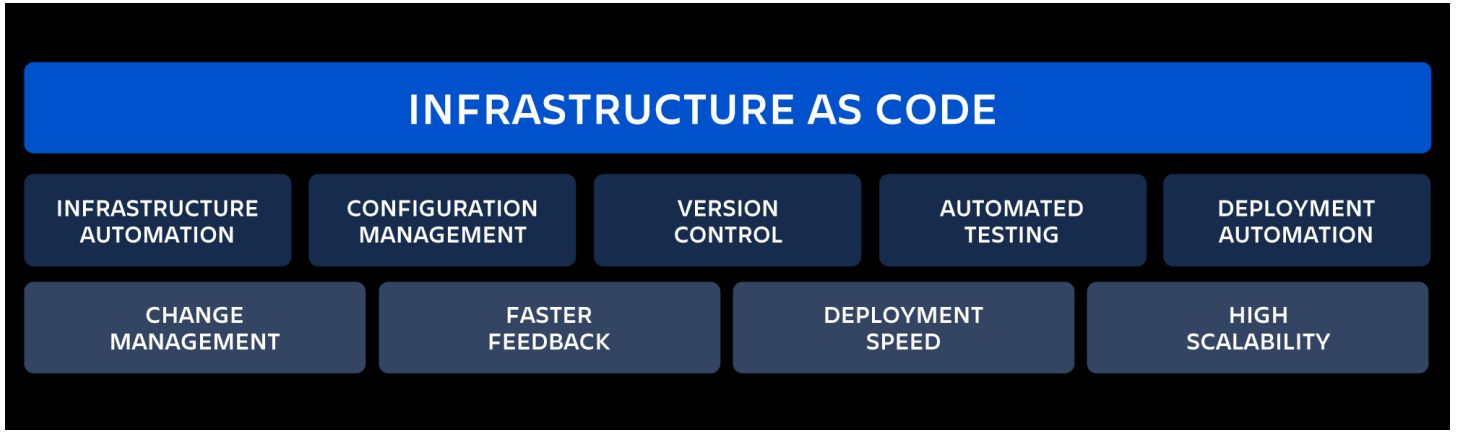
Брайан Доусон (Cloudbees)

3. DevOps

Як вже зазначалося, в минулому існував чіткій розподіл обов'язків: розробники писали програмний код, а команди системних адміністраторів відповідали за налаштування обладнання й операційних систем, які на ньому працювали.

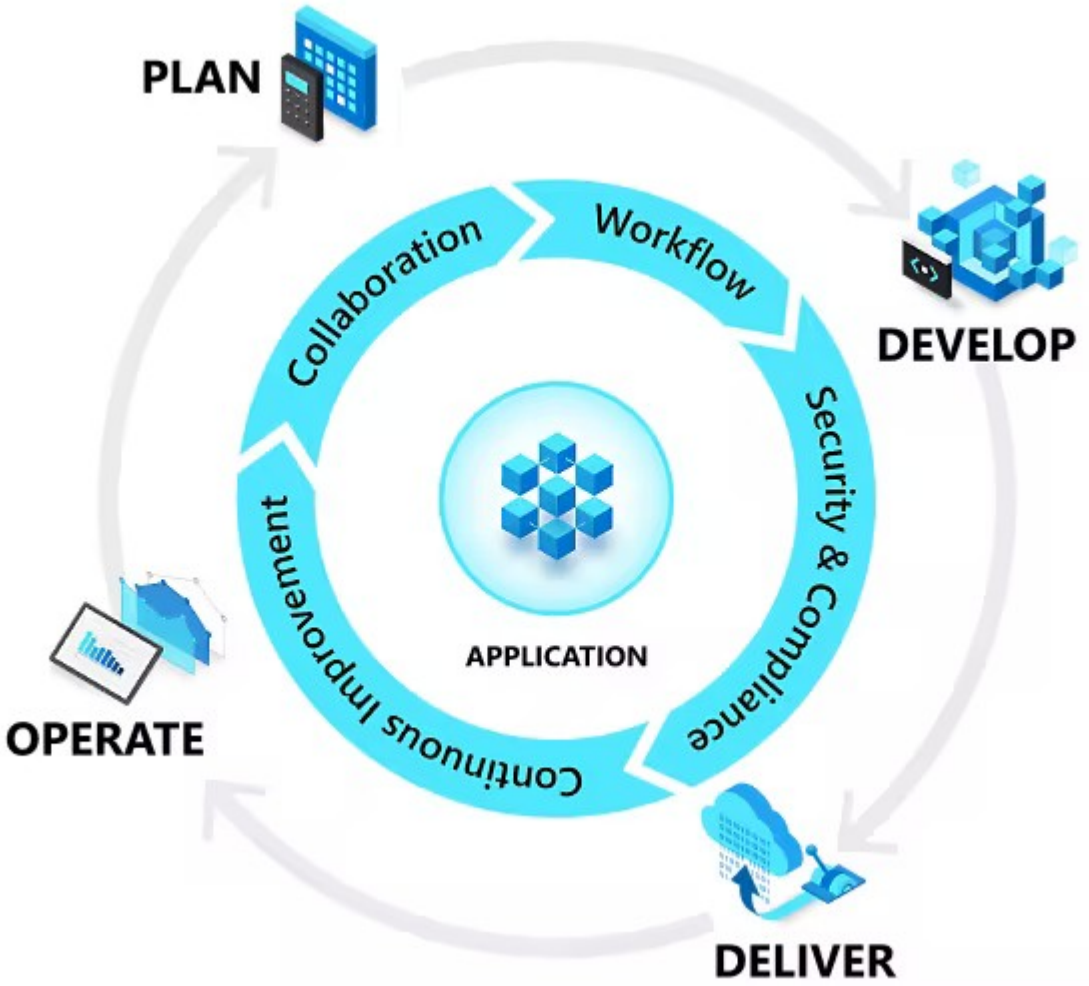
Сьогодні, коли у багатьох випадках обладнання розташоване у хмарі, сісадмінівські операції перейшли з фізичного рівня (наприклад, встановлення ОС або прокладання та налаштування мережі) на програмний: адміністратори тепер змушені писати програмний код для налаштування обладнання (автоматизації хмари).

Ця концепція отримала назву **інфраструктура як код** (IaC – Infrastructure as Code).



3. DevOps

Життєвий цикл розробки програм із застосуванням DevOps можна представити так:



4. Контейнери

Для практичного впровадження складного програмного забезпечення зазвичай потрібно встановити й велику кількість його додаткових залежностей:

- бібліотеки;
- інтерпретатори або компілятори;
- розширення;
- мовні пакети;
- ...

Крім того, зазвичай слід налаштовувати відповідну конфігурація:

- ліцензійні ключі;
- файлові системи;
- паролі доступу до баз даних;
- гарячі клавіші;
- ...

4. Контейнери

Деякі сучасні мови програмування пропонують спеціалізований інструментарій для управління конфігураціями (пакетні менеджери), наприклад, **cargo** в мові програмування **Rust**; **eggs** у **Python**; **gems** у **Ruby** і т. п. На жаль, ці засоби працюють лише в межах певної мови і в цілому не завжди вирішують проблему залежностей.

Ще одним способом практичного впровадження програм зі всіма їхніми залежностями є використання **віртуальних машин**, коли користувач фактично отримує цілу комп’ютерну систему, необхідну для запуску програми у вигляді образу віртуальної машини. Проте такий підхід має багато недоліків: віртуальна машина має великий об’єм, потребує багато часу та дискового простору для розгортання, складна в обслуговуванні, повільно завантажується та розгортається, а також дуже неефективна з точки зору продуктивності та вживаних системних ресурсів.

4. Контейнери

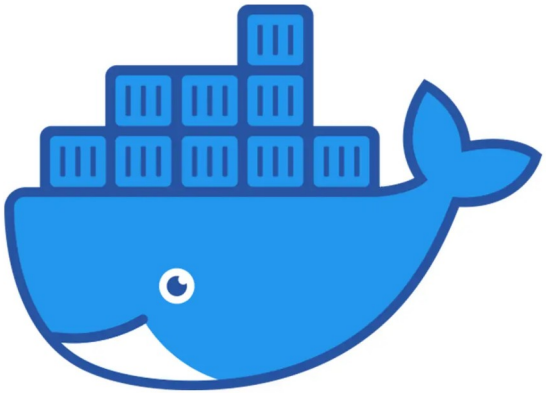
Для розв’язання вищезазначених проблем ІТ-галузь запозичила в індустрії вантажного судноплавства **концепцію контейнерів**: у 1950-х роках американський далекобійник **Малкольм МакКлін** (Malcolm McLean) запропонував наступну ідею для пришвидшення вантажних перевезень: замість вивантаження товарів з вантажівок, в яких вони були доставлені до морського порту, а потім перенесення їх на кораблі і навпаки, краще просто завантажити на кораблі причепи вантажівок (контейнери).

4. Контейнери

Програмні контейнери засновані на тій самій, що і морські, ідеї – це стандартний універсальний і широко поширений формат пакування й доставлення, який дозволяє суттєво збільшити пропускну спроможність, знизити грошові витрати, заощадити на масштабуванні та спростити обслуговування.

Формат контейнеру передбачає можливість зберігання всього, що необхідно для його роботи, додаючи це в єдиний файл образу, який може бути запущений відповідним **середовищем виконання контейнерів**.

На відміну від файлу **образу віртуальної машини** (ВМ) контейнер не містить нічого зайвого. Крім того, контейнери працюють безпосередньо на реальному процесорі (**шар віртуалізації відсутній**), що істотно підвищує швидкість роботи контейнерів у порівнянні з ВМ. Розмір контейнерів значно менший за розмір образів ВМ (на декілька порядків).



5. Kubernetes

Компанія **Google** почала широко використовувати контейнери з промисловими навантаженнями раніше за всіх. Більшість сервісів Google давно працюють в контейнерах (**Gmail**, **Google Search**, **Google Maps**, **Google App Engine** тощо).

Для автоматизованого управління спільною роботою великої кількості контейнерів (**оркестрації**) Google у 2014 р. розробила спеціальну програмну платформу з відкритим початковим кодом **Kubernetes** (κυβερνήτης – керманич).

Kubernetes займається тим самим, що й найкращі системні адміністратори: автоматизацією, централізованим веденням журналу, моніторингом, забезпеченням відмовостійкості. Ця платформа бере те, чому ми навчилися у спільноті DevOps і робить з цього готове рішення за замовчуванням.

Келсі Хайтауер (Google)

kubernetes

5. Kubernetes

Багато традиційних задач системного адміністрування (оновлення серверів, встановлення патчів безпеки, налаштування мереж, створення резервних копій і т. п.) не є найбільш пріоритетнішими у хмарно-орієнтованому середовищі. Платформа Kubernetes здатна автоматизувати всі ці дії, щоб ваша команда могла зосередитись на виконанні основної роботи.

Частина з цих можливостей (наприклад, автомасштабування або балансування навантаження) вбудовані у ядро Kubernetes, інші постачаються у вигляді різноманітних розширень, які використовують стандартне **API Kubernetes**.

Використання Kubernetes дає можливість знизити витрати на на інфраструктуру та більш ефективно використовувати наявні ресурси.

Важливою особливістю Kubernetes є його незалежність від хмарного провайдера.

6. Дорожня мапа DevOps

Дорожня мапа вивчення DevOps має наступний вигляд (https://roadmap.sh/devops)