DevOps

Áttekintés

- Software Development (Dev) + Information Technology Operations (Ops)
- Célja a szoftver fejlesztésének és üzemeltetésének összehangolása.
- 3-szintű párbeszéd:
    - **Emberek:** Az emberek közti kommunikációs nehézséget a legfontosabb leküzdeni.
    - **Folyamatok:** A jó kommunikáció mellett is szükséges, hogy mindenki jól lássa saját helyét és felelősségét a folyamatban, hatékony legyen az együttműködés.
    - **Eszközök:** A DevOps nem egy termék, amit megvehetünk, de fontosak az eszközök, amivel megvalósítjuk.
- Az üzlet agilitása megkívánja a gyors reakciót és a csapat hatékony működését.
    - Gyakori kiadások
    - Lehetőleg kevés drága emberi munkával
    - Hibalehetőségek csökkentésével
    - Tipikus megoldás az **automatizálás**

Fejlesztő feladatai

- Meetingeken való részvétel
- Tesztek írása: specifikáció megfogalmazása programkóddal
- Funkciók implementálása: specifikáció megvalósítása programkóddal
- Tesztek futtatása
- Közös kód fordítása
- Telepítés
- Telepítés ellenőrzése
- Értesítés telepítésről

Problémák automatizálás hiányában

- Minden **merge** előtt célszerű megbizonyosodni arról, hogy merge után az alkalmazás helyesen, specifikáció szerint fog működni.
- Minden **push** előtt célszerű lenne megbizonyosodni az alkalmazás helyességéről.
- Kézzel végezve ez sok, repetitív munka, ami előbb-utóbb könnyen elmaradhat.
- Egy fejlesztő ismeri egyedül a folyamatot és annak egy részét: fluktuáció esetén a tudás elveszik, átadása másik ember felé költséges és hiányos lehet.
- A folyamat minden egyes eleméhez szaktudás szükséges: fejlesztői és rendszergazdai ismeretek megléte.
- Ha éppen nem elérhető a fejlesztő, akkor nem lesz új release.

DevOps jellemzői

A rendszerünk, jellemzően naponta egyszer-kétszer hajtja végre a teljes folyamatot.

- **Integrált:** Minden változtatás, művelet egy láncként hajtódik végre.
- **Lefordított:** A kód lefordul és termék keletkezik, például futtatható állomány.
- **Tesztelt:** Automatizált tesztek lefutnak.
- **Archivált:** Verzióval ellátott és tárolt, szükség esetén felhasználható.
- **Élesített:** Feltöltődik abba a célkörnyezetbe, ahol a fejlesztők használhatják.

A hatékony működés elemei

Build

- **Gyakori commitolás**
  - Naponta egyszer-kétszer, átlátható feladat méret.
  - Így hamar kiderül, ha valami nem működik.
  - A csapattagok hamar értesülnek a változásokról.
  - Elkerülhető a kiadás előtt álló összes hatalmas merge, amik kockázatosak és nehezen becsülhetőek.
- **Automatizált és gyors build folyamat**
  - Külön gépen, nem lassítva a fejlesztői gépet.
  - A fejlesztő tud dolgozni, míg a folyamat zajlik.
  - Akkor segíti a munkát, ha hamar kapunk visszajelzést.
- **Kódelemzés riport**

Teszt

- **Öntesztelő build**
  - Fordítás után automatikusan fussanak le a tesztek.
  - Ellenőrzi, hogy a szoftver úgy viselkedik-e, ahogy a fejlesztő elvárja.
- **Gyors tesztek**
  - Akkor segíti a munkát, ha hamar kapunk visszajelzést.
  - Akár könnyen visszaállhatunk egy korábbi verzióra.
- **A tesztelés fusson üres, független gépen**
  - „Nálam megy” jelenség elkerülése.
  - Hamar észlelhetjük, ha a fejlesztői környezet eltér a tesztkörnyezettől.
  - Az integrációs hibák hamar kiderülnek valódi tesztkörnyezetben.

Telepítés

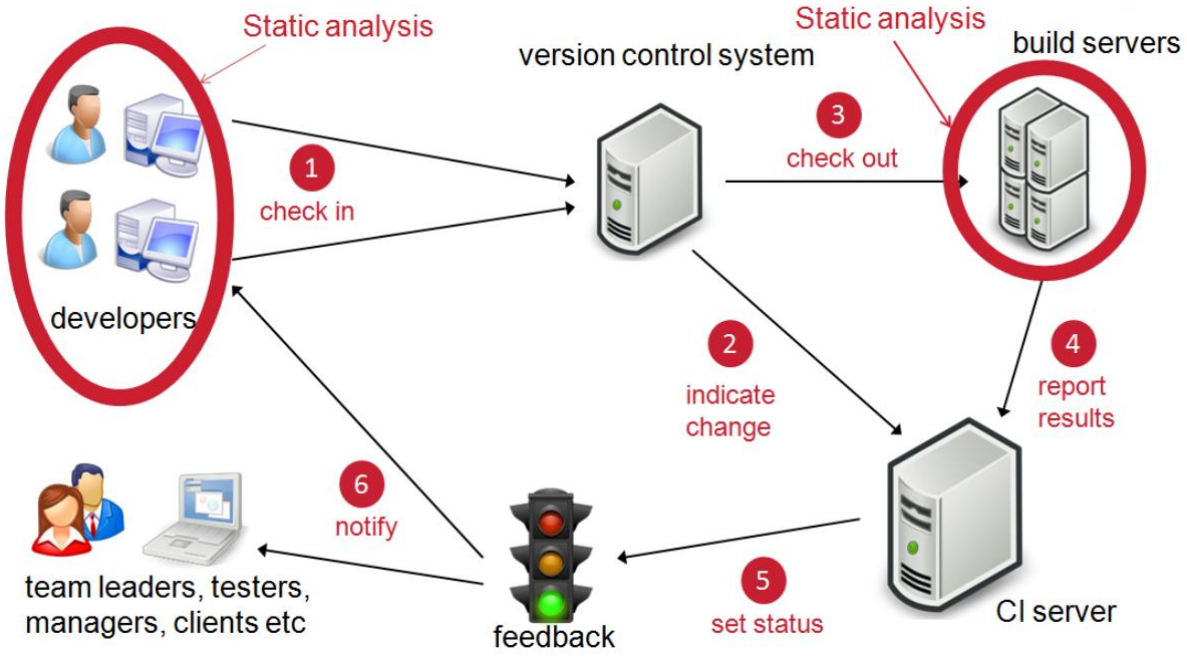
- **Automatikus telepítés**
  - Script által futtatott telepítés.
  - Éles rendszerekbe nem automatikusan, de néha automatizáltan telepítünk.

- **Legyen könnyen elérhető a legutóbbi build**
  - Ezáltal a mindenkori legfrissebb verzió tesztelhetővé, demózhatóvá válik.
  - Akár a menedzser is felhasználhatja.
- **Mindenki lássa a buildek eredményeit**
  - Közösen tudjuk vizsgálni a szoftver állapotát.
  - Könnyen felderíthető egy sikertelen build és a hiba oka is.
- **Jól követhető a szoftver állapota és alakulása**

Continuous Integration (CI)

Áttekintés

- **A nem-fejlesztői feladatokat nem a fejlesztőnek kell végeznie.**
  - Több idő marad a fejlesztésre.
  - Felelősséget leveszi a fejlesztő válláról.
- **A szoftver állapotának automatikus monitorozása.**
  - Több projekt állapotának párhuzamos figyelése.
  - A fejlesztés mélyebb, részletesebb követése.
  - Trendek figyelése, változtatások visszamérése.
- **Azonnali visszajelzés**
  - A fejlesztőnek.
  - A megadott csapattagoknak, például vezető fejlesztőnek.
  - A vezetőségnek.
  - A megrendelőnek.
- **Bármikor elérhető teszt verzió**



Lehetőségei

- Integráció verziókezelőkkel.
- Teszt riportok generálása és elküldése.
- Mentés különféle **artifact repository**-ba:
    - **Artifact:** a build folyamatok eredménytermékei.
    - A kimenetek automatikus mentése.
- Élesítés különböző környezetekbe.
- A projekten dolgozók értesítése:
    - Checkin-ről
    - Sikeres vagy sikertelen build-ről
    - Telepítésről
    - Tesztelés eredményéről
- Egyéb eszközökkel való integrálás (pl. ANT, Maven, Jenkins)
- Célja, hogy a teljes szoftverfejlesztési folyamat, amely emberi interakciót is igényel, sorrendiséget követel meg, egyetlen nagy, összefüggő, **automatizált flow**-vá váljon.

CI-rendszer

Elemei

- Maga a CI-t megvalósító komponens (aki futtatja a build scripteket) – adatbázis, ahol tárolja a konfigurációkat
- Verziókezelő, ahol elérhető a forráskód
- Fordító
- Kódelemző
- Az artifactory (artifact repository), ahol tárolja a kész termékeket
- Az alkalmazásszerver, ahová telepít
- A ticket kezelő rendszer, ami kezeli az értesítéseket és hibajegyeket
- Az eszköz használói:
    - Fejlesztők
    - CI-szakember
    - Stakeholder-ek (vezetőség és megrendelők)

Választási szempontok

- Elterjedtség: közösség és fejlesztői támogatás
- Kompatibilitás: támogatott nyelvek és platformok
- Árazás és licenszelés: a hostolás is költséges
- Repository-k támogatása
- Értesítési módok

Continuous Delivery (CD)

Áttekintés

- A CI természetes kiterjesztése: olyan megközelítés, amelyben a csapatok biztosítják a rendszer gyors kiadhatóságát verziónként.
- Célja, hogy minél előbb visszajelzést kapjunk a munkánkról, hogy azt minél tökéletesebben a megrendelő igényeihez igazíthassuk, és ezzel minél nagyobb elégedettséget érjünk el a megrendelőknél.

A teljes folyamat

- **Continuous Integration + Continuous Delivery**
- Használati esetek:
    - **Használjuk teszt szervereken**
        - Egy speciális tesztelő kör számára hozzáférhető.
        - Ha hibás, a cég akkor se veszít üzleti presztízséből.
    - **Ne használjuk éles szerveren és kliensen**
        - Új verzióról értesíteni kell a felhasználókat, amint összeszedetten célszerű: napi több frissítés már zavaró és túl sok változást nehéz feldolgozni.
        - Egy kiadás jellemzően néhány hetes fejlesztési funkciókat tartalmaz, amit már teszteltek a tesztelők és a megrendelők.
        - Ezt **mindig kézzel** csináljuk és alaposan ellenőrizzük, amint lehetséges.
- Előnyök:
    - Változtatási kérelem esetén **push** után a CI-rendszer ezt **érzékeli**, automatikusan **futtatja a teszteket** és amennyiben minden sikeres, **telepíti** is a saját teszt szerverünkre.
    - Ez nem éles rendszer, hanem a sajátunk – ha bármi tönkremegy, javítjuk.
    - Ráadásul ez tényleg gyorsan kell, akár óránként többször is, az ügyfél úgysem látja.
    - Az ügyfél teszt szerverét is kezelhetjük **remote branch**-en.