# Szoftverfejlesztői karrier alapjai

**Főbb tartalmi elemek:**

* **Karrierindítás és tanulás**
  + Hogyan válassz programozási nyelvet és technológiát.
  + Tanulási stratégiák, önálló projektek és portfólióépítés.
  + Junior pozíció megszerzésének lépései.
* **Szakmai fejlődés**
  + Kódolási készségek fejlesztése, tervezési minták, tesztelés és hibajavítás.
  + Soft skillek: kommunikáció, csapatmunka, prezentációs készség.
  + Hogyan válj megbízható és értékes csapattaggá.
* **Karrierépítés és váltás**
  + Munkakeresés, önéletrajz-készítés, állásinterjúkra felkészülés.
  + Fizetéstárgyalás, előléptetés kiharcolása.
  + Freelancer munka, vállalkozás indítása vagy távmunka lehetőségei.
* **Személyes fejlődés és motiváció**
  + Időgazdálkodás, fókuszálás, produktivitás.
  + Folyamatos tanulás és naprakészen tartás a gyorsan változó iparágban.
  + Mentális egészség, kiégés megelőzése, munka–magánélet egyensúly.

A könyv gyakorlati tanácsokat, motivációs tippeket és konkrét akcióterveket ad, hogy a fejlesztők tudatosan és hosszú távon építsék a karrierjüket.

## **Szoftverfejlesztői karrier alapjai** **– Fejezetenkénti összefoglaló**

### **1. Bevezetés**

* A könyv célja, hogy útmutatót adjon a teljes fejlesztői karrierhez, a kezdőtől a tapasztalt szakemberig.
* John Sonmez hangsúlyozza a folyamatos tanulás és a tudatos karrierépítés fontosságát.

### **2–6. Az alapok**

* Hogyan válassz programozási nyelvet és technológiai irányt.
* A “learning by doing” módszer előnyei – saját projektek, portfólió.
* Kódolás mellett a problémamegoldó gondolkodás fejlesztése.

### **7–11. Technikai fejlődés**

* Haladó kódolási technikák, tiszta kód elvei, tervezési minták.
* Verziókezelés, tesztelési módszerek, hibakeresés.
* Technológiai trendek követése és alkalmazkodás a változásokhoz.

### **12–16. Soft skillek és kommunikáció**

* Hatékony kommunikáció kollégákkal, ügyfelekkel és vezetőkkel.
* Csapatmunka, konfliktuskezelés, prezentációs készség.
* Hogyan tűnj ki a munkahelyeden mint megbízható szakember.

### **17–21. Munkakeresés és interjúk**

* Önéletrajz és online profil (LinkedIn, GitHub) optimalizálása.
* Interjúfelkészülés: technikai kérdések, viselkedéses kérdések.
* Hogyan szerezz ajánlásokat és használd a kapcsolati hálódat.

### **22–26. Fizetés és előléptetés**

* Fizetési tárgyalás stratégiák, bértárgyalási pszichológia.
* Előléptetés kiharcolása teljesítménnyel és értékteremtéssel.
* Munkahelyváltás előnyei és kockázatai.

### **27–31. Alternatív karrierutak**

* Freelancer munka előnyei és nehézségei.
* Vállalkozás indítása fejlesztőként.
* Távmunka lehetőségek és globális munkaerőpiac.

### **32–36. Személyes fejlődés**

* Időgazdálkodás és produktivitási technikák (Pomodoro, batching).
* Motiváció fenntartása, célkitűzés.
* Munka–magánélet egyensúly és kiégés megelőzése.

### **37–Vége. Záró gondolatok**

* A karrier egy hosszú távú befektetés.
* A folyamatos fejlődés és alkalmazkodás a siker kulcsa.
* Ne csak kódolj – építs kapcsolatokat és értéket teremts.

**Kezdj egy működő alkalmazás megvizsgálásával**

A legtöbb kezdő programozó, amikor tanulni akar, elővesz egy könyvet, és elkezdi olvasni. Bár vannak nagyszerű, gyakorlatorientált könyvek, szerintem a legjobb kiindulópont az, ha megnézed egy valóban működő alkalmazás forráskódját, és megpróbálsz minél többet megérteni belőle.

Ez nehéz lesz.  
 Kényelmetlennek fogod érezni, de ez rendben van. Szokj hozzá ehhez az érzéshez.

Válassz ki egy nyílt forráskódú alkalmazást — lehetőleg egy népszerűt, amely valószínűleg jól meg van tervezve —, és kezdd el böngészni a forráskódját. Sok ilyen projektet találsz a GitHub-on, így érdemes ott keresni.

Ha le tudod tölteni, lefordítani és futtatni is az alkalmazást, az még jobb. Ha van barátod, aki segíthet ebben, az nagyszerű; ha nincs, az sem baj. A lényeg, hogy ismerkedj a kóddal: figyeld meg, hogyan néz ki a programozási nyelv szintaxisa, és próbáld megérteni, mi mit csinál.

Ha lehetséges, használd magát az alkalmazást is, így látni fogod az összefüggést a kód és a működés között. Eleinte lehet, hogy úgy érzed, semmit sem értesz — ez teljesen rendben van. A cél, hogy megpróbálj legalább egy-két dolgot megfejteni, vagy kitalálni, mit változtatnál a kódban, hogy módosítsd a működését.

Figyeld meg a neveket, a szervezést, és próbálj úgy viselkedni, mintha egy régész lennél, aki egy ősi civilizáció írását próbálja megfejteni.

Ezzel a módszerrel komoly előnyt szerzel azokkal szemben, akik úgy akarnak megtanulni programozni, hogy még nem is látták, hogyan néz ki az adott nyelv kódja. Mint minden utazás előtt, itt is érdemes először feltérképezni a terepet — a programozás sem különbözik ettől.

# Azure DevOps for Web Developers

Lépésről lépésre, hogyan lehet a **Microsoft Azure DevOps** eszközeit hatékonyan használni a modern szoftverfejlesztési életciklusban.

### **Főbb tartalmak:**

* **Azure DevOps alapjai** – Az eszközkészlet áttekintése: Azure Repos, Pipelines, Boards, Test Plans és Artifacts.
* **Forráskód-kezelés** – Git és Team Foundation Version Control (TFVC) használata, ágazási (branching) és összeolvasztási (merging) stratégiák.
* **CI/CD folyamatok** – Build- és release-pipeline-ok létrehozása, konfigurálása, automatizálása, valamint alkalmazások telepítése Azure-ba és más környezetekbe.
* **Agilis projektmenedzsment** – Scrum és Kanban támogatása Azure Boards segítségével, backlog-kezelés, sprinttervezés, feladatkövetés.
* **Tesztelés és minőségbiztosítás** – Automatizált tesztek, unit- és integrációs tesztek beépítése a pipeline-okba, manuális tesztelési lehetőségek.
* **Csomagkezelés és verziózás** – Artefaktok és NuGet-csomagok kezelése Azure Artifacts használatával.
* **Biztonság és jogosultságkezelés** – Felhasználók, csapatok és jogosultsági szintek menedzselése.

### **A könyv célja:**

* Segíteni a **webfejlesztőket** abban, hogy teljesen ki tudják használni az Azure DevOps integrált eszközeit.
* Bemutatni, hogyan lehet **automatizált, biztonságos és skálázható** fejlesztési és üzemeltetési folyamatokat kiépíteni.
* Átvezetni az olvasót az ötlet megszületésétől a kódoláson, a tesztelésen és a telepítésen át a folyamatos karbantartásig.

### **Röviden:**

A könyv egy teljes körű **DevOps kézikönyv webfejlesztőknek**, amely gyakorlati példákon keresztül mutatja meg, hogyan lehet a fejlesztési folyamat minden lépését automatizálni és optimalizálni az **Azure DevOps** segítségével.

### **Bevezetés**

A DevOps az egész IT-spektrum egyik alapvető eleme, amely egyesíti az embereket, a folyamatokat és a technológiát azzal a céllal, hogy gyorsabban szállítsunk értéket. Ugyanakkor a DevOps számos változatban létezik, eltérő szintű megértéssel, ami sok megvalósítás számára akadályt jelenthet.

Ebben a könyvben a DevOps különböző koncepcióit fogjuk felfedezni, és elindítunk téged egy DevOps-utazáson az Azure DevOps segítségével. Megvizsgáljuk a DevOps megvalósításának teljes folyamatát az Azure DevOps eszközein keresztül.

Kifejezetten az alábbi témákat érintjük:

* Projektmenedzsment, beleértve a felhasználó- és jogosultságkezelést
* Követelménykezelés
* Verziókezelés
* Tesztmenedzsment
* Build automatizálás és kiadáskezelés
* Tesztmenedzsment (ismétlődő pont az eredetiben)
* Folyamatos visszajelzés

Emellett kitérünk a jelenlegi IT-környezetre, és arra, hogyan tudjuk a DevOps előnyeit különböző területeken – például támogatási projektekben – kihasználni. Minden fogalmat több nézőpontból mutatunk be: kezdő, DevOps fejlesztői, DevOps architekt és üzleti szemszögből.

Ez a könyv elsősorban a DevOps-ban kezdőknek szól, akik szeretnék megismerni a DevOps fogalmait a modern alkalmazások teljes spektrumán. Ugyanakkor elegendő információt nyújt a DevOps mérnökök számára is, hogy finomítsák tudásukat a DevOps különböző területein.

## **Tartalomjegyzék**

**1. fejezet: A DevOps alapjai és változatai**

* Szoftverfejlesztés
* DevOps
* DevOps csapat
* DevOps gyakorlatok
* DevOps változatok
* Előnyök
* Azure DevOps
* Összefoglalás

**2. fejezet: Projektmenedzsment az Azure DevOps használatával**

* Szervezetek
* Szervezet létrehozása
* Szervezeti beállítások
* Projektek
* Projekt létrehozása
* Projektbeállítások
* Folyamat-sablon testreszabása
* A testreszabás megkezdése
* Új munkatétel hozzáadása
* Összefoglalás

**3. fejezet: Követelménykezelés az Azure DevOps használatával**

* Munkatételek
* Munkatételek
* Szűrők
* Nézetbeállítások
* Opciók
* Táblák
* Tábla kiválasztása
* Analitika
* Nézetbeállítások
* Tábla beállításai
* Kártyák
* Backlogok
* Oszlopbeállítások
* Opciók
* Sprintek
* Lekérdezések
* Új lekérdezés
* Offline munka
* Összefoglalás

**4. fejezet: Verziókezelés az Azure DevOps használatával**

* Repos
* Ágkezelés és összeolvasztás
* Projekt repó
* Ágak és tagek
* Commitok
* Pushok
* Ágak
* Tagek
* Pull requestek
* Munka a Visual Studio-val
* Munka a Visual Studio Code-dal
* Munka Git Bash-sel
* Mintaalkalmazás
* Angular alkalmazás verziókezelése
* .NET API verziókezelése
* Összefoglalás

**5. fejezet: Tesztmenedzsment az Azure DevOps használatával**

* Tesztesetek
* Megosztott lépések és paraméterek
* Tesztcsomagok
* Teszttervek
* Define fül
* Execute fül
* Diagram
* Előrehaladási jelentés
* Paraméterek
* Konfigurációk
* Futtatások
* Terhelésteszt
* Összefoglalás

**6. fejezet: Build automatizálás és kiadáskezelés**

* Build- és release-folyamat
* Folyamatos integráció
* Folyamatos kézbesítés és telepítés
* Pipeline-ok
* Pipeline létrehozása
* Klasszikus szerkesztő
* Környezetek
* Release-ek
* Release pipeline létrehozása
* Könyvtár
* Változócsoportok
* Biztonságos fájlok
* Feladatcsoportok
* Telepítési csoportok
* Mintamegvalósítás buildre és release-re
* Angular alkalmazás buildje és release-e
* .NET alkalmazás buildje és release-e
* Összefoglalás

**7. fejezet: Folyamatos visszajelzés és egyéb funkciók**

* Irányítópultok
* Wiki
* Kód közzététele Wikiként
* Wiki szerkesztése
* Megbeszélések vagy hozzászólások
* Felhasználói beállítások
* Folyamatos visszajelzés
* Azure Application Insights
* Összefoglalás

**8. fejezet: DevOps architekturális sablonok**

* A DevOps megközelítés webalkalmazásokhoz
* Mikroszolgáltatások
* A DevOps megközelítés adatbázisokhoz
* SQL Server adatbázis
* A DevOps megközelítés gépi tanulási modellekhez
* Gépi tanulás operacionalizálása
* Azure DevOps
* A DevOps megközelítés kész (COTS) alkalmazásokhoz
* A DevOps megközelítés a támogatási csapat számára
* Intelligens swarming támogatási modell
* Összefoglalás

**Tárgymutató**

## **1. fejezet: A DevOps alapjai és változatai**

**Ambily K K** Hyderabad, India

A szoftverfejlesztési folyamatok és a kiadási követelmények időről időre változhatnak annak érdekében, hogy a vállalatok továbbra is értéket nyújtsanak az ügyfeleknek, és piaci elfogadottságot érjenek el.

A méretezhetőség és a korai visszajelzés megszerzése érdekében számos szervezet gyakori kiadási módszertant alkalmaz az új funkciók vagy hibajavítások bevezetésére. Ezek a kiadások olyan gyakorisággal kerülhetnek éles rendszerbe, mint hetente egyszer, de előfordulhat, hogy naponta, óránként, vagy akár tíz másodpercenként is.

Annak érdekében, hogy a vállalatok megőrizzék piaci előnyüket vagy gyors visszajelzési ciklust alakítsanak ki, egyre inkább bevezetik a DevOps gyakorlatokat, mint például a folyamatos kézbesítést (Continuous Delivery) és a folyamatos telepítést (Continuous Deployment).

Ebben a fejezetben bevezetést kapsz a DevOps fogalmaiba és gyakorlataiba.

### **Szoftverfejlesztés**

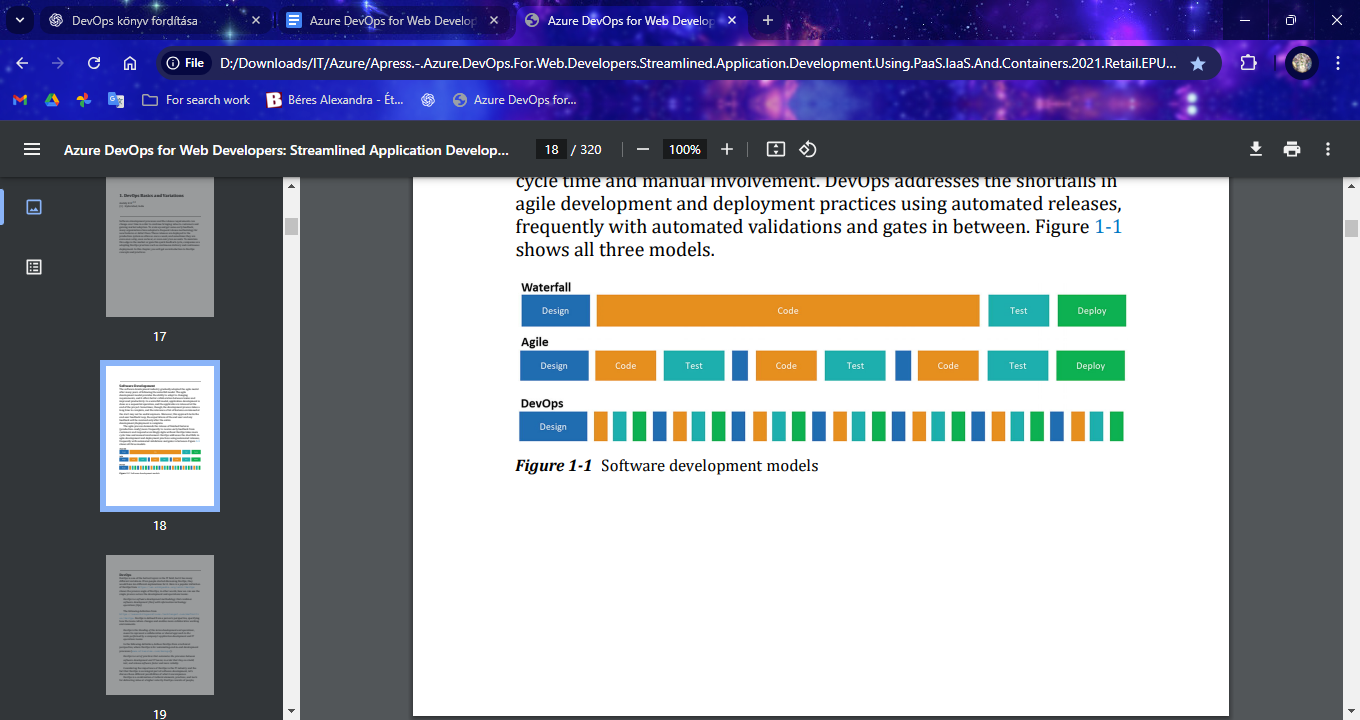
A szoftverfejlesztési iparág hosszú éveken keresztül a vízesés modellt követte, majd fokozatosan áttért az agilis modell alkalmazására. Az agilis fejlesztési modell lehetővé teszi, hogy a csapatok könnyebben alkalmazkodjanak a változó követelményekhez, emellett elősegíti a jobb együttműködést a különböző csapatok között, és növeli a termelékenységet.

A vízesés modellben az alkalmazásfejlesztés szigorúan egymást követő lépésekben zajlik, és a teljes alkalmazás csak a projekt végén kerül kiadásra. Ez azonban gyakran azzal jár, hogy a fejlesztési folyamat nagyon hosszú időt vesz igénybe, és mire a projekt befejeződik, sok olyan funkció, amelyet eredetileg elképzeltek, már elveszíti az aktualitását, vagy egyszerűen nem hasznos a felhasználók számára. További probléma, hogy ebben a megközelítésben hiányzik a végfelhasználói visszajelzés folyamata: a felhasználók elvárásai és visszajelzései csak a teljes fejlesztési és telepítési folyamat lezárása után jutnak el a fejlesztőkhöz.

Az agilis módszertan ezzel szemben megköveteli, hogy a kész funkciók – vagyis a gyártásra kész részek – jóval gyakrabban kerüljenek kiadásra. Ennek célja, hogy a felhasználók minél korábban adhassanak visszajelzést, amelyre a fejlesztőcsapat időben tud reagálni. Ugyanakkor, ha az agilitást nem egészíti ki a DevOps szemlélet, a fejlesztési ciklus még mindig hosszabb időt vehet igénybe, és sok manuális munkát igényel.

A DevOps éppen ezeket a hiányosságokat kezeli az agilis fejlesztési és telepítési gyakorlatokban: lehetővé teszi az automatizált kiadásokat, gyakran beépített automatikus ellenőrzésekkel és ellenőrzőpontokkal a különböző lépések között.

Az 1-1. ábra a három szoftverfejlesztési modellt mutatja be.



### **DevOps**

A DevOps napjaink egyik legfelkapottabb témája az IT világában, ugyanakkor számos különböző értelmezése és változata létezik. Ha tíz ember elkezdene beszélgetni a DevOps-ról, jó eséllyel tíz eltérő magyarázatot adnának rá.

Az alábbiakban néhány népszerű definíciót láthatunk:

* **A folyamatközpontú nézőpontból** a Wikipédia (<https://en.wikipedia.org/wiki/DevOps>) így fogalmaz:  
    
   *„A DevOps egy olyan szoftverfejlesztési módszertan, amely egyesíti a szoftverfejlesztést (Dev) az informatikai üzemeltetéssel (Ops).”*
* **Az emberi tényezőre összpontosítva** a TechTarget meghatározása (https://searchitoperations.techtarget.com/definition/DevOps) szerint:  
    
   *„A DevOps a development (fejlesztés) és az operations (üzemeltetés) szavak összeolvadásából született, és azt a közös vagy együttműködésen alapuló megközelítést jelenti, amelyet a vállalat alkalmazásfejlesztési és IT-üzemeltetési csapatai a feladatok végrehajtására használnak.”*
* **A technikai szemszögből** az Atlassian ([www.atlassian.com/devops](http://www.atlassian.com/devops)) így határozza meg:  
    
   *„A DevOps olyan gyakorlatok összessége, amelyek automatizálják a szoftverfejlesztés és az IT-csapatok közötti folyamatokat annak érdekében, hogy gyorsabban és megbízhatóbban tudjanak szoftvert építeni, tesztelni és kiadni.”*

Figyelembe véve a DevOps fontosságát az IT iparban, valamint azt a tényt, hogy a DevOps a szoftverfejlesztés szerves részévé vált, érdemes részletesebben is áttekinteni, mi mindent foglalhat magában.

A DevOps a **kultúra, a gyakorlatok és az eszközök** kombinációja, amelynek célja, hogy gyorsabb ütemben biztosítson értéket. A DevOps három alappillére: **emberek, folyamatok és technológia**.

* A technológia lehetővé teszi a végponttól végpontig tartó szállítási folyamat automatizálását.
* A megfelelő eszközök használata elősegíti a csapatok közötti együttműködést.
* Az agilis módszertan biztosítja a rugalmasságot a végrehajtásban, a csapatok közötti együttműködést, az átfogó tevékenységmenedzsmentet és az innovatív csapatkultúrát.

A **DevOps kultúra** azt is jelenti, hogy a csapattagok többféle készségben kapnak képzést, így szélesebb körű rálátásuk lesz az egész folyamatra. Ez elősegíti a csapatok közötti gördülékeny együttműködést, az „egységes csapat” szemlélet kialakulását és a több területen jártas munkatársak meglétét.

A legtöbb vállalat a DevOps bevezetésekor a technológiai és folyamatbeli megvalósításra összpontosít. Számukra az **agilis fejlesztés** az alapértelmezett módszertan, az **automatizálás** pedig a DevOps bevezetésének fő mozgatórugója.

A **DevOps kultúra** azonban összetettebb kérdés: nemcsak új folyamatokat és eszközöket igényel, hanem szervezeti struktúraváltást, valamint a munkatársak gondolkodásmódjának és hozzáállásának változását is.

### **DevOps Csapat**

Ha a DevOps a fejlesztés és az üzemeltetés keveréke, akkor mi a helyzet a teszteléssel foglalkozó részleggel és más érintettekkel?

A válasz az, hogy minden érintettnek megvan a maga fontossága és szerepe a DevOps-ban.

A **„Dev”** a DevOps-ban nem csupán a fejlesztőket jelenti, hanem ide tartoznak a **scrum masterek, fejlesztők, tesztelők, projektmenedzserek, szakterületi szakértők (SME-k), üzleti elemzők, domain szakértők, biztonsági architektek, megoldás-architektek, teljesítménymérnökök** és így tovább. A szerepek és felelősségek projektről projektre eltérhetnek, attól függően, hogy a megoldás milyen méretű, mennyire összetett, és mekkora üzleti hatással bír.

Az **„Ops”** a DevOps-ban az **üzemeltetési szerepköröket** fedi le, például a build mérnököket, az éles rendszert támogató szakembereket, a monitorozással foglalkozókat stb.

Fontos megjegyezni, hogy a DevOps célja **nem** a költségcsökkentés vagy a folyamatok automatizálása – ezek csupán melléktermékei a megfelelő DevOps gyakorlatok bevezetésének. A DevOps igazi célja az, hogy **értéket szállítson az ügyfél számára**.

A csapatstruktúra kulcsfontosságú szerepet játszik abban, hogy a DevOps gyakorlat mennyire lesz sikeres egy szervezeten belül. Különféle paraméterek alapján meghatározható, hogy adott projekt esetén melyik csapattopológia a leginkább megfelelő.

Érdemes megvizsgálni, hogy a csapatod **helyes DevOps csapatstruktúrát** követ-e, vagy esetleg valamelyik **anti-típus** irányába tartozik. Erről részletesen olvashatsz Matthew Skelton cikkében:  
 👉<https://web.devopstopologies.com/>

### **DevOps Gyakorlatok**

A DevOps különböző elemeket foglal magában annak érdekében, hogy egy teljes, végponttól végpontig tartó DevOps munkafolyamatot lehessen megvalósítani. Az alábbiakban a legfontosabb gyakorlatok közül említünk néhányat:

* **Követelménykezelés (Requirement management):** Az összes követelmény nyomon követhetőségének biztosítása, gyors áttekintés lehetősége, valamint az érvényesség és a használhatóság szerinti prioritások kezelése a követelmények állapotában.
* **Konfigurációkezelés (Configuration management):** A konfigurációk kezelése a célkörnyezetek, a felhasználók, a vállalati szabályzatok és egyéb tényezők alapján.
* **Kiadáskezelés (Release management):** A kiadások megtervezése, ütemezése és felügyelete, beleértve a jóváhagyásokat és más szabályzatokat is.
* **Folyamatos integráció (Continuous integration):** A legújabb változtatások beolvasztása a távoli tárházakba (repository-kba), valamint annak biztosítása, hogy a minőségi követelményeknek megfelelő build-ek jöjjenek létre.
* **Folyamatos telepítés (Continuous deployment):** A kódváltoztatások telepítése magasabb szintű környezetekbe, miután a build minőségét automatizált ellenőrzési pontokkal igazolták.
* **Infrastruktúra mint kód (Infrastructure as code):** Az infrastruktúra biztosítása a kódtelepítés részeként, valamint a konfigurációk kezelése szkript formátumban.
* **Tesztautomatizálás (Test automation):** A tesztek automatizálása annak érdekében, hogy gyorsabb végrehajtás és szélesebb körű lefedettség biztosítható legyen.
* **Folyamatos monitorozás (Continuous monitoring):** Olyan rendszerek alkalmazása, amelyek automatikusan figyelik és kiértékelik az eredményeket. Ez magában foglalhatja a hibák naplózását, a helyesbítő intézkedések megtételét vagy értesítések küldését.

Ezeken túl számos egyéb terület is kapcsolódik a DevOps gyakorlathoz, például az együttműködés, a tudásmenedzsment, az identitás- és jogosultságkezelés, a portfóliómenedzsment, az artefaktum-tárházak kezelése, a teljesítménytesztelés, a biztonsági tesztek, a backlog-kezelés és sok más.

### **DevOps Változatok**

A DevOps a szoftverfejlesztés szerves részét képezi. A legtöbb projekt legalább a DevOps technikai és folyamatbeli részeit implementálja, még akkor is, ha ez önmagában nem hozza meg a DevOps valódi előnyeit. A **kulturális változások** azok, amelyek valóban lehetővé teszik a DevOps teljes értékének kiaknázását. Gyakran azonban a DevOps-ot csupán az **automatizálásra és költségcsökkentésre** szorítjuk, ahelyett, hogy az értékteremtést helyeznénk előtérbe.

Jelenleg számos DevOps-változat létezik az implementáció módjától függően. Néhány jelentősebb változat:

* **BizDevOps:** A DevOps egyik változata, amely a **üzleti szempontok bevonását** hangsúlyozza a teljes folyamat során. Bár a hagyományos DevOps is számol az üzleti érintettek részvételével, a BizDevOps kiemeli az üzleti szereplők aktív szerepét a teljes végrehajtásban.
* **DevSecOps / SecDevOps:** Ez a változat **biztonságot integrál a DevOps folyamatokba**, és a jelenlegi trendek szerint a legtöbb vállalat ezt az modellt alkalmazza.  
    
   Miért van szükség a biztonságra? Bár a kódunkban már megvalósítottunk bizonyos biztonsági megoldásokat, a DevOps folyamat részeként is biztosítani kell a biztonságot. Néhány példa, ahol a hagyományos DevOps-ban biztonsági kockázatok léphetnek fel:  
  + Több eszközt használó DevOps-implementáció esetén szükség van az eszközök közötti integrációra jelszavak, titkos kulcsok, tokenek segítségével, amelyeket biztonságos tárolókban kell kezelni.
  + Ha CI/CD pipeline-t használunk, amely a kódcommitokat és a telepítést automatikusan végzi, hogyan biztosítható, hogy az éles környezetbe kerülő kód biztonságos? Ezt automatizált biztonsági ellenőrző tesztekkel valósítjuk meg.
  + A DevOps „shift left” elve szerint miért kellene várnunk egy build-re, hogy feltérképezzük a biztonsági sérülékenységeket? Integrálhatunk-e IDE-t biztonsági funkciókkal a kódolás közben? Igen, ezt például **security IntelliSense-szel** tehetjük meg.
  + Hogyan biztosítható egy felhőalapú előfizetés? Ezt **előfizetésbiztonságnak** (subscription security) nevezzük.
  + Az éles környezetben a biztonsági problémák figyelése és kezelése. A konténerekben telepített alkalmazások biztonsági monitorozása további biztonsági réteget igényel, ezt **hosting environment/container security**-nek nevezzük.
* A **SecDevOps** célja a DevOps teljes körű biztonsági igényeinek kezelése. A megfelelő biztonsági eszközök integrálhatók a DevOps életciklusba a megoldás technológiája, a hosting környezet és a DevOps eszközök alapján.
* **CloudOps:** Ez a DevOps egyik változata, amely kifejezetten a **felhőalapú rendszerekre** fókuszál. Fő különbség a **„infrastruktúra mint kód”** megvalósítása és a SaaS/PaaS telepítések kezelése.
* **MLOps:** A DevOps elengedhetetlen komponens a **gépi tanulási (ML) modellek fejlesztésében**. Az MLOps implementációk némileg eltérnek a hagyományos felhőalapú alkalmazásfejlesztésektől. Az MLOps folyamat főbb lépései:  
  + Adatok importálása különböző forrásokból
  + Adatfeldolgozás (átalakítás, tisztítás stb.)
  + Modell betanítása mintadaton
  + Modell telepítése
  + Valódi adatok feldolgozása, esetleges újratanítás a teljes orchestration részeként
* Az MLOps-ot speciális DevOps eszközökkel kezelik, például **MLFlow** és **Azure ML Pipeline** segítségével.
* **Intelligens DevOps:** Ez a DevOps jövője, amely a **DevOps és a mesterséges intelligencia** együttműködésén alapul. A DevOps fő három komponense: **emberek, folyamatok és technológia**.  
    
   A DevOps életciklus során rengeteg adat keletkezik, de sok esetben ezek nincsenek megfelelően felhasználva. Az intelligens DevOps a gyűjtött adatokat – például logokat, kódokat, teszteseteket, hibákat, build-jelentéseket – elemzi, és ezekből értékes, előrejelzési célú információkat nyer. Például:  
  + Telepítési hibák előrejelzése
  + Minőségi kockázatok előrejelzése
  + Új tesztscenáriók azonosítása
  + Hibák előrejelzése logelemzés alapján

A DevOps folyamat az idők során fejlődött: **DevOps 1.0-tól az intelligens DevOps-ig**.