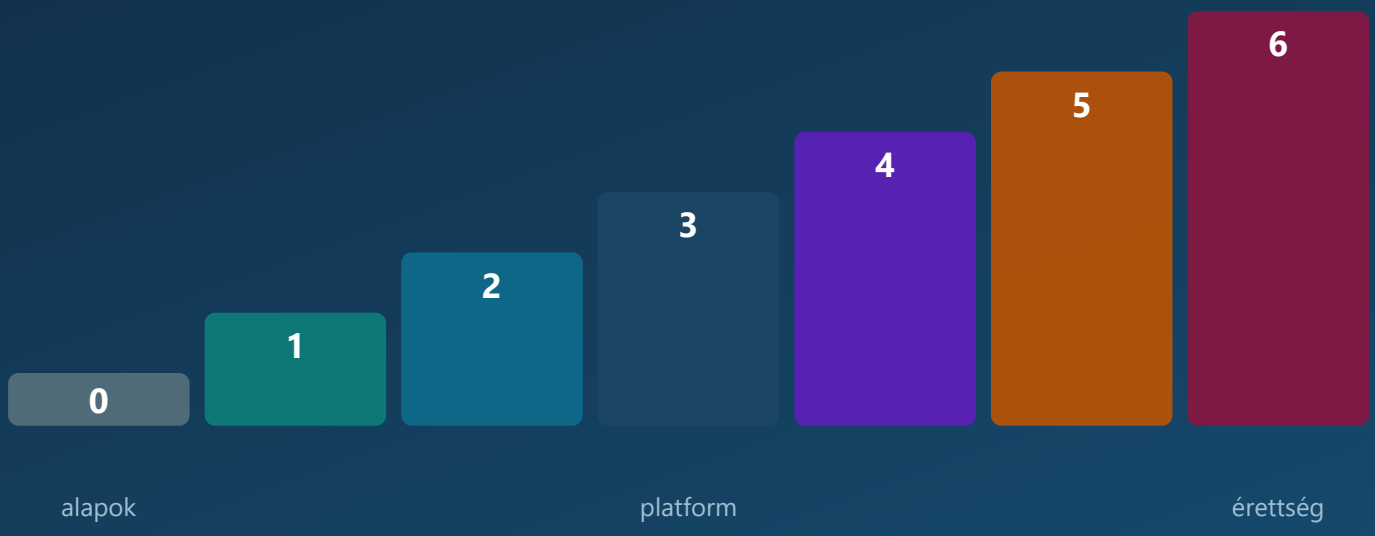
DevOps bevezetés a gyakorlatban

Sorrend, függőségek, kommunikáció —
így áll össze rendszerré a DevOps egy cégben



Puskás Balázs

DevOps

2026
Belső kiadás

DevOps bevezetés a gyakorlatban

Függőség-alapú forgatókönyv az atomi elemektől a komplex platformig

Írta: **Puskás Balázs**

Kapcsolat: Mobil: +36 30 680 0800 · E-mail: puskas.balazs@36306800800.hu

Tartalom

ELŐSZÓ

Miért készült ez a könyv

Kevés hálátlanabb feladat van, mint egy félbemaradt DevOps-bevezetést rendbe tenni. Az eszközök megvannak, a számla ki van fizetve, a rendszer mégsem áll össze — mert a darabok rossz sorrendben kerültek a helyükre. Ez a könyv ennek a sorrendnek a forgatókönyve.

A könyv tanácsadói munka tapasztalatából született. A minta cégről cégre ugyanaz: van Kubernetes, de kézzel is belenyúlnak; van CI, de a titkok a repositoryban laknak; van monitoring, mentés viszont nincs. Egyik hiba sem az eszközök hibája. A rendszer hiányzik köréjük — és ez mindig a legrosszabbkor, az első komoly incidensnél derül ki.

Két olvasónak készült. Az egyik a mérnök, aki holnap reggel elkezdené felépíteni mindezt, és tudni akarja, mihez nyúljon először. A másik a vezető, akinek mindezt jóvá kell hagynia — neki azt mutatja meg, mit kérjen számon, és miből látja, hogy megéri.

Fontos tisztázni az elején, mire vállalkozik ez a könyv. Forgatókönyvet ad, nem kézikönyvet: megmutatja, mi mire épül és miért, de nem helyettesíti az egyes eszközök részletes dokumentációját. A platform, amelyre a forgatókönyv épül, végig ingyenes és nyílt forráskódú marad — fizetni legfeljebb akkor kell, ha egy audit gyártói támogatást követel —, és on-prem környezetre készült, felhős kitekintéssel.

Végül egy tanács az olvasáshoz. Az első két fejezet a fogalmakat és a célt rögzíti, a negyedik a kommunikációt. A könyv gerince azonban a harmadik fejezet, a függőség-alapú bevezetési sorrend — aki csak egyetlen fejezetre szán időt, az ezt olvassa el.

Puskás Balázs · DevOps

1. FEJEZET

A DevOps mint működési modell

> *A DevOps nem eszközlista és nem munkakör, hanem az a mód, ahogy a szoftver a fejlesztő gépétől az éles üzemig eljut — mérhetően, ismételhetően, auditálhatóan.*

FOGALOM

DevOps — a fejlesztés (*Development*) és az üzemeltetés (*Operations*) szavak összevonása. Eredetileg mozgalomként indult azzal a céllal, hogy a két, hagyományosan elkülönült terület közös felelősségben, közös eszközökkel dolgozzon; mára gyűjtőfogalom, amely a szállítás és az üzemeltetés teljes működési modelljét jelöli.

Ebben a könyvben a DevOps a következőt jelenti: a szoftver fejlesztése, kiadása és üzemeltetése **egyetlen folyamat, amely automatizált, mérhető és auditálható**. Minden változás verziókezelt, a biztonság a folyamatba van építve, a visszacsatolás folyamatos. Az eszközök, amelyekkel ezt elérjük, idővel cserélődni fognak — a működési elvek nem.

FOGALOM

CI/CD — Continuous Integration / Continuous Delivery. A CI a kód automatikus fordítása, tesztelése és ellenőrzése minden változtatásnál; a CD a kész, ellenőrzött változat automatizált eljuttatása az éles környezetig. A kettő együtt azt garantálja, hogy az élesítés ismételhető, automatizált rutinfolyamat, nem kézi beavatkozás.

A működés hét pillére

1. **Verziókezelés és kollaboráció** — minden változás (kód, infrastruktúra-leírás, konfiguráció, dokumentáció) Gitben él, kötelező szakmai átnézéssel (code review). Ami nincs Gitben rögzítve, az nem követhető és nem auditálható.
2. **CI** — automatizált build, teszt és biztonsági ellenőrzés minden változtatásra, programnyelvenként szabványosított futószalagokkal (pipeline).
3. **GitOps-alapú szállítás** — a futtatókörnyezet kívánt állapota is Gitben van; egy szoftverkomponens élesítése nem kézi parancs, hanem egy jóváhagyott módosítás beolvasztása. Visszavonás = a módosítás visszavonása.
4. **Infrastructure as Code** — a szerverek, hálózati elemek, klaszterek és jogosultságok kódból jönnek létre. A kézi, dokumentálatlan módosítás (ClickOps) tiltott, mert nem megismételhető és nem követhető.
5. **Identitás és titokkezelés** — egyetlen központi bejelentkezés (SSO) minden eszközhöz, központi titoktár, szabályozott belépő-kilépő folyamat. Kilépéskor a hozzáférés egyetlen központi helyen vonható vissza, nem rendszerenként külön-külön.
6. **Megfigyelhetőség és célszámok** — metrikák, naplók, hívásláncok egy helyen; a megbízhatóság számokban kifejezve (SLO, hibaköltségvetés), nem megérzésben.
7. **Üzemeltetési fegyelem** — runbookok, incidenskezelés, változáskezelés, mentés és katasztrófa-helyreállítás, és az ezekhez tartozó kommunikáció.

Mindezeken keresztbe fekszik a **biztonság és a megfelelőség**: nem külön projekt, hanem a pillérekbe épített kontrollok összessége. Egy jól felépített DevOps-működés az audit-bizonyítékok nagy részét melléktermékként termeli.

> **FOGALOM**
>
> **GitOps** — az az elv, hogy a rendszerek kívánt állapota deklaratívan, verziókezelve, Gitben van leírva, és egy automata (a bemutatott platformban az ArgoCD) folyamatosan ehhez az állapothoz igazítja a valóságot. Következmény: a Git-történet egyben teljes változásnapló — ki, mit, mikor, kinek a jóváhagyásával.

Szereplők és felelősségük

Szereplő	Fő felelősség
Vezetőség (CEO/CTO/CFO)	Irány, erőforrás, kockázatvállalási döntések; a kibervédelmi szabályozás (NIS2) szerinti vezetői felelősség és képzés.
DevOps / platform mérnök	A platform (CI/CD, Kubernetes, megfigyelhetőség, IaC) építése és üzemeltetése; a fejlesztők kiszolgálása és edukálása.
Fejlesztő	Az alkalmazáskód és annak üzemeltethetősége: életjel-végpontok (health probe), strukturált naplók, metrikák, erőforrás-igények megadása.
Üzemeltető / technikus	Napi felügyelet, runbook-végrehajtás, első reagálás, eszkaláció.
Biztonsági felelős	Kockázatkezelés, sérülékenység-javítási határidők, incidensbejelentés a hatóság felé.
Compliance/audit felelős	Bizonyítékgyűjtés, audit-felkészülés, szabályzatok karbantartása.
Hálózat / hardver (netops)	Fizikai infrastruktúra, VLAN-ok, tűzfal; a DevOps definiált, írásos interfészen kér tőlük változást.

A felelősségek folyamatonkénti pontos leosztását (ki dönt, ki csinál, kit kell megkérdezni, kit kell tájékoztatni) RACI-mátrixban érdemes rögzíteni még a bevezetés elején — ez a 3. fejezet 0. szintjének egyik eleme.

> **FOGALOM**
>
> **RACI** — felelősségi mátrix: minden folyamathoz megmondja, ki a *Responsible* (végrehajtó), *Accountable* (felelős döntéshozó — mindig pontosan egy), *Consulted* (megkérdezendő) és *Informed* (tájékoztatandó). A mátrix előzetes rögzítése megelőzi a tisztázatlan felelősségből adódó fennakadásokat.

Meddig ér a DevOps? — a teljes tématérkép

Hogy mi tartozik a DevOps-hoz, arra nincs hivatalos válasz — de van jobb módszer a találgatásnál. Ha öt egymástól független forrást fektetünk egymásra (DORA, roadmap.sh és CNCF, Google SRE, ITIL 4 és ISO 27001, AWS/Azure Well-Architected), több mint hetven téma rajzolódik ki, hét nagyobb területre rendezve:

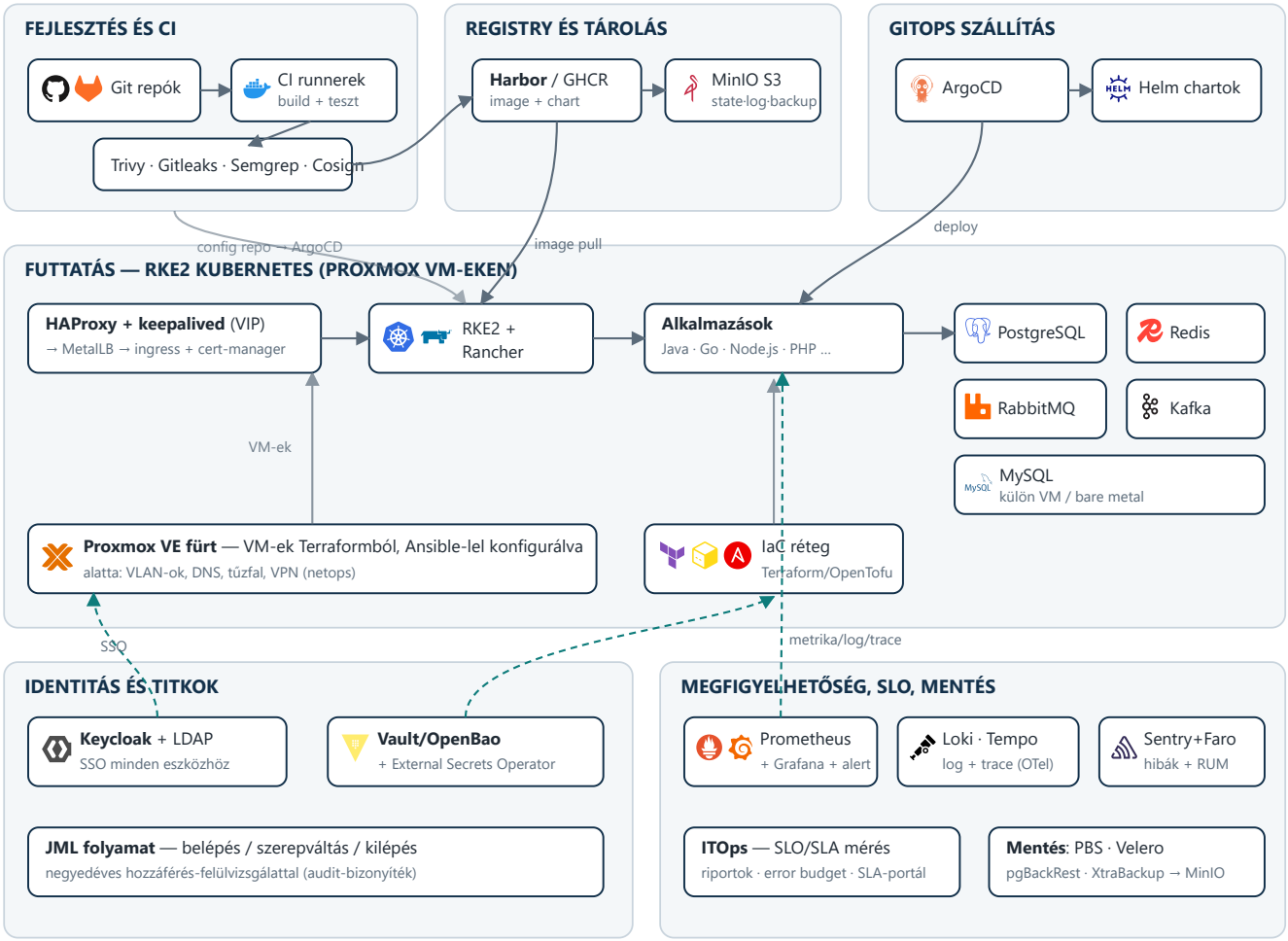
Terület	Ide tartozik többek között
Szoftverszállítás	verziókezelés, CI, tesztelés, release stratégiák, séma-migráció, registry
Infrastruktúra és platform	IaC, virtualizáció, Kubernetes, hálózat, DNS és TLS, skálázás, kapacitás
Megfigyelhetőség és mérés	metrika, napló, trace, riasztás, SLO/SLA, költség-láthatóság
Megbízhatósági gyakorlatok	mentés és DR (katasztrófa-helyreállítás), üzletmenet-folytonosság, túlterhelés-kezelés, helyreállítási próbák
Biztonság	supply chain, titokkezelés, identitás, sérülékenység-kezelés, peremvédelem — és a fizikai meg az emberi réteg is
Megfelelés és governance	NIS2, ISO 27001, audit-bizonyítékok, eszköz- és beszállító-nyilvántartás
Szervezet és kommunikáció	szerepek és felelősségek, fejlesztő-edukáció, vezetői riport, ügyeleti kultúra

Ennyi témával egyetlen bevezetés sem indul, és nem is kell. A térkép arra való, hogy ami kimarad, az döntésből maradjon ki, ne feledékenységből: a könyv a hét terület magját fedi le, a többit a függelék tartja számon.

2. FEJEZET

A célarchitektúra

> *Egy forgatókönyvet könnyebb követni, ha látjuk a végét. Ez a fejezet a kész platform képét mutatja meg: az elemeket és a köztük futó kapcsolatokat.*

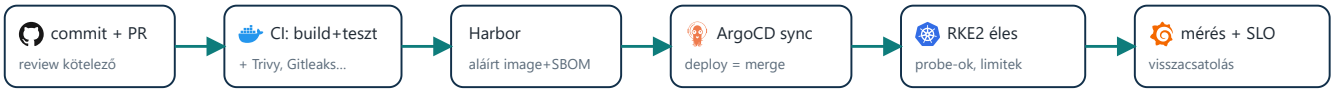


1. ábra — A célarchitektúra: a hat zóna és a fő adatáramok

Egy változás útja a committól az élesig

1. A fejlesztő módosítási kérelmet (pull request) nyit → szakmai átnézés + automatikus ellenőrzések: fordítás, tesztek, biztonsági szkennelés → beolvasztás.
2. A CI megépíti a konténer-image-et, aláírja, összetevő-leltárt (SBOM) készít róla, és a Harbor registrybe tölti; ezután a konfigurációs repositoryban átírja az új verziószámot.
3. Az ArgoCD észleli a konfigurációváltozást, és a klasztert az új kívánt állapothoz igazítja — ez maga az élesítés. Kézi parancs nincs.
4. Az induló alkalmazás a titkait a központi titoktárból kapja (External Secrets); a forgalom a HAProxy VIP → MetalLB → ingress útvonalon éri el az alkalmazást.

5. Futás közben metrika a Prometheusba, napló a Lokiba, hívási lánc a Tempóba, alkalmazáshiba a Sentrybe kerül; az ITOps ezekből SLA-teljesülést és havi riportot számol.
6. Minden lépés bizonyítékot termel — a pull request, a pipeline-napló, a szinkronizálás története együtt a teljes, auditor által is elfogadható változásnapló.



2. ábra — A szállítási lánc: minden lépés automatikus és nyomon követhető

Hol fut mi?

Réteg	Komponensek
SaaS / külső	GitHub (vagy gitlab.com), Let's Encrypt tanúsítványok
Proxmox VM — Kubernetesen kívül	Self-hosted GitLab (ha ez a választás), MySQL, HAProxy+keepalived pár, Proxmox Backup Server, Sentry, opcionálisan a titoktár külön VM-eken
RKE2 klaszter	ArgoCD, Harbor, Keycloak, megfigyelhetőségi stack, ITOps, RabbitMQ/Kafka, PostgreSQL (operátorral), maguk az alkalmazások
Bare metal (opció)	Nagy IO-igényű MySQL, storage node-ok

Környezetek

Minimum három környezet kell: **dev** (fejlesztői játszótér), **staging** (az éles tükörképe — ide kerül minden az éles előtt), **prod** (csak a GitOps útvonalon változtatható). Kisebb szervezetnél a dev és a staging összevonható; az éles környezet elkülönítése minden esetben kötelező.

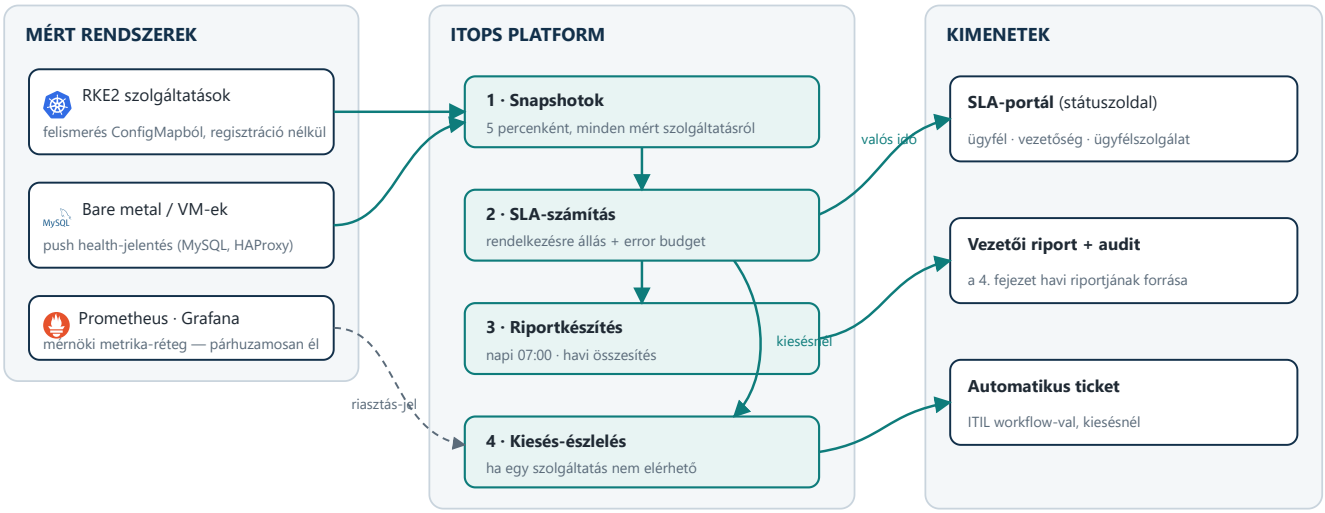
> **FOGALOM**
>
> **SLI · SLO · SLA** — a mérőszám, a cél és a szerződés. Az SLI a mért érték (pl. a sikeres kérések aránya), az SLO a magunknak kitűzött cél (pl. 99,9% havonta), az SLA az ügyfélnek tett, következményekkel bíró ígéret (pl. 99,5% + kötbér). A kettő közti rés a mozgástér: az SLO mindig szigorúbb, mint az SLA.

SLO/SLA/SLI mérés a gyakorlatban: az ITOps platform (mlops.hu) — opcionális

A metrikagyűjtés önmagában még nem SLA-mérés. A Prometheus nyers számokat ad; ahhoz, hogy ezekből szerződés szintjén értelmezhető kimutatás, error budget-követés és bizonyító erejű riport készüljön, külön mérési rétegre van szükség. Ezt a szerepet a platformban az **ITOps** (mlops.hu) tölti be: self-hosted, Kubernetes-natív üzemeltetési platform, amely a szolgáltatások rendelkezésre állását méri, követi és riportolja. A réteg opcionális — a stack nélküle is teljes, de az SLA-riportok, a státuszoldal és a kiesési ticketek kézi munkáját váltja ki.

A platform képességei:

- **Valós idejű SLA-monitoring.** A platform ötpercenként snapshotot készít a szolgáltatások állapotáról, ebből számítja a rendelkezésre állást, és szolgáltatásonként vezeti az error budgetet.
- **Automatikus riportok.** Minden reggel hét órakor napi, a hónap zárásakor havi SLA-jelentés készül, beavatkozás nélkül. Ez adja a vezetői riport rendelkezésre állási sorát és az auditon bemutatható bizonyítékot.
- **Automatikus service discovery.** A mérendő Kubernetes-szolgáltatásokat ConfigMapokból ismeri fel, kézi regisztráció nélkül. Amikor új szolgáltatás kerül élesbe, a mérés magától bővül vele.
- **SLA-portál.** Önálló státuszoldal mutatja a valós idejű rendelkezésre állást a vezetőségnek, az ügyfeleknek és az ügyfélszolgálatnak. Incidens és karbantartás idején ez a hivatalos tájékoztatási felület.
- **ITIL-kompatibilis ticketing.** Szolgáltatás-kiesésnél a platform automatikusan ticketet nyit, vizuális workflow-szerkesztővel. Az incidens dokumentálása így nem múlik emberi fegyelmen.
- **Bare metal támogatás.** A Kubernetesen kívül futó gépek — például a külön VM-en működő MySQL vagy a HAProxy-pár — push-alapú health-jelentéssel kapcsolódnak be ugyanabba a mérésbe.



3. ábra — Az ITOps működése: a mért rendszerektől a snapshotokon és az SLA-számításon át a státuszoldalig, a riportokig és az automatikus ticketig

A platform négy okból illik kiemelten ebbe a stackbe:

- **Kubernetes-natív, és Helmmel telepíthető.** Percek alatt a meglévő RKE2 klaszterre kerül, külön infrastruktúrát nem igényel.
- **Teljesen self-hosted.** Az adatok a cégnél maradnak, nincs felhőfüggés és nincs felhasználónkénti díj — ez on-prem környezetben és az adatkezelési megfelelőség szempontjából egyaránt előny.
- **Ingyenes Community szinttel indul.** Ez illeszkedik a stack alapelvéhez; az SLA-riport és a ticketing modul a Professional, illetve Enterprise szint része **[FIZETŐS]** — a mérési alap és a service discovery a Community szinten is működik.
- **A mérést, a riportot és a kommunikációt egyetlen rendszerbe köti.** Az SLA-mérés, a státuszoldal és a kiesési ticket egy helyen él, ezért a 4. fejezetben leírt vezetői riport és incidens-tájékoztatás közös, ellentmondásmentes forrásból dolgozik.

Prometheus + Grafana (mérési alap)	ITOps (SLA-réteg)
Nyers metrikák gyűjtése, technikai riasztások, hibakeresési dashboardok	Szerződésszintű rendelkezésre állás, error budget, napi és havi riport, státuszoldal, automatikus ticket
Célközönsége a mérnöki csapat	Célközönsége a vezetőség, az ügyfél és az auditor — valamint a mérnökök SLO-döntései

3. FEJEZET

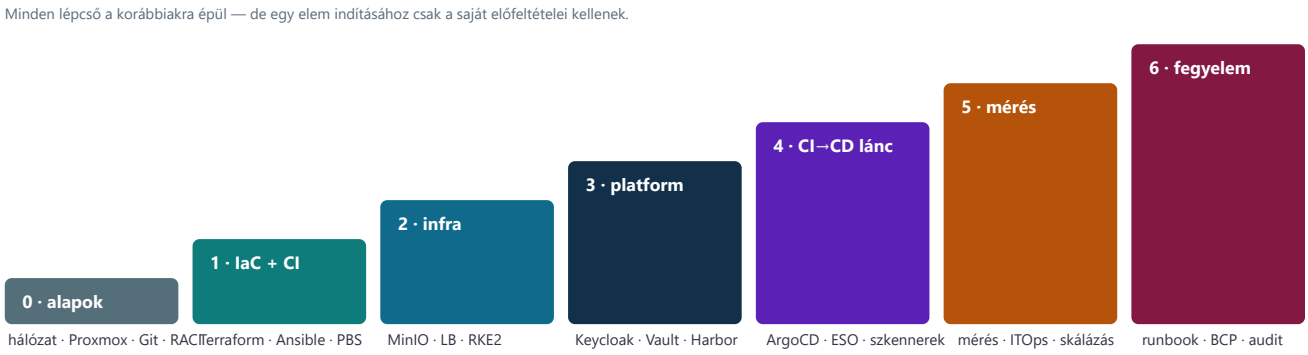
A bevezetés forgatókönyve — mi mire épül

> *Egyetlen nagy lista, amelyben minden elemnél az áll, mi az előfeltétele. Nem naptár és nem projektterv: a sorrendet a függőség adja, nem a hónapok.*

Aki bevezetési tervet készít, rendszerint ütemtervet vár: mi történik az első hónapban, mi a másodikban. Itt mást talál. Ennek a listának minden soránál az áll, hogy az adott elemnek mi az előfeltétele, mert egy szervezetben valójában ez dönti el a sorrendet, nem a naptár. Egy elem akkor kezdhető el, amikor az előfeltételei készen vannak; aminek nincs közös függősége, az akár párhuzamosan is haladhat. Az út az atomi, semmire nem épülő elemektől vezet a **komplexek** felé.

A sorrend meghatározásának szabálya

1. Azok az elemek kezdhetők el, amelyeknek minden előfeltétele teljesül — ezek az aktuálisan elvégezhető munkák.
2. Ami egy szinten van, tetszőleges sorrendben vagy párhuzamosan végezhető.
3. Nem kell egy szintet befejezni a következő megkezdéséhez: elég, ha az adott elem *saját* előfeltételei készek. (A Velero mentés indítható, amint van Kubernetes és MinIO — akkor is, ha a Keycloak még sehol.)
4. A szemlélet-jellegű elemek (runbook-írás, IaC-fegyelem, fejlesztő-edukáció) nem „a végén jönnek”: az első naptól érvényesek, csak az eszközparkjuk épül ki fokozatosan.



4. ábra — A hét szint: az atomi elemektől az érettségig

0. szint Atomi elemek — semmire nem épülnek

Elem	Előfeltétel	Mit tesz lehetővé
Hálózati alapok: IP/VLAN terv, tűzfal-szabályok, DNS zónák, VPN	— (a netops csapattal közös)	minden on-prem elem elhelyezését
✖ Proxmox fürt (3+ node, storage)	hálózati alapok	minden VM-et és a Kubernetest

Git platform + repo-szabályzat, branching	— (SaaS esetén semmi)	minden kód, konfiguráció, IaC tárolását — **minden más ide commitol**
Feladatkezelő (az ügyfél meglévője) + Confluence tudásbázis	—	a munka és a runbookok követhetőségét
Szerepkörök, RACI, kommunikációs ritmus rögzítése	—	hogy mindenki tudja, ki miért felel

1. szint Az alapokra épül

Elem	Előfeltétel	Mit tesz lehetővé
Terraform/OpenTofu + Ansible alap IaC *(a state először lokálisan/Gitben, később MinIO-ba költözik)*	Proxmox + Git	minden további VM-et kódból
CI alap-pipeline-ok (build + teszt, felhős runnerekkel)	git	a nyelvi pipeline-okat, image buildet
Proxmox Backup Server	Proxmox	VM-szintű mentést — **minél korábban, annál jobb**
LDAP/AD felmérés vagy OpenLDAP kiépítés	— / IaC	a Keycloak-federációt

2. szint IaC-ből épített alap-infrastruktúra

Elem	Előfeltétel	Mit tesz lehetővé
MinIO VM-ek (verziózás + object lock)	IaC	Terraform state, naplók, trace-ek, mentések tárolását
HAProxy + keepalived LB-pár (VIP-ek)	IaC + IP-terv	a K8s API és az ingress elé a belépési pontot
MySQL VM (ha az alkalmazások igénylik)	IaC	a bare metal adatbázis-réteget
RKE2 klaszter (3 vezérlősík + N worker, CIS-profil, etcd-mentés MinIO-ba) + alap K8s-biztonság (RBAC, Pod Security, NetworkPolicy)	IaC-ből VM-ek + portnyitások + LB a 6443-ra	minden Kubernetes-alkalmazást

> **FOGALOM**
>
> **Kvórum — miért 3 node?** Két gépnél a rendszer vita esetén nem tudja eldönteni, melyik fél a hibás — mindkét oldal működőnek tekintheti magát (split-brain). Három gépnél többségi döntés születik: kettő mindig többség. Ezért 3 a minimum a vezérlősíkra, nem 2 — a harmadik gép a folyamatos működés feltétele.

3. szint A klaszterre épülő platform-szolgáltatások

Elem	Előfeltétel	Mit tesz lehetővé

MetalLB + ingress controller (Traefik) + cert-manager	RKE2 + IP-pool + DNS wildcard	hogy bármi kívülről elérhető legyen TLS-sel
Rancher	RKE2	klaszter-kezelő felületet, RBAC-ot
PostgreSQL (CloudNativePG operátor)	RKE2	a Keycloak, Harbor, ITOps adatbázisait
Vault/OpenBao titoktár (+ audit log)	RKE2 vagy 3 VM	minden titok központi tárolását
Keycloak (LDAP-federációval)	RKE2 + PostgreSQL (+LDAP)	az SSO-t **minden** eszközhöz + a belépő-kilépő folyamatot
Harbor registry (Trivy-szkenneléssel, MinIO-tárolással)	RKE2 + MinIO + ingress	a saját image- és charttárolást, proxy cache-t
Redis (ha az alkalmazások igénylik)	RKE2	cache/session réteget

A SZABÁLY MINTÁJA — A „VAULT-PÉLDA"

Amíg a titoktár (Vault) nincs kész, nincs External Secrets. Amíg nincs External Secrets, az ArgoCD-vel telepített alkalmazások titkai nem megoldottak. Tehát: **Vault → External Secrets → GitOps-alapú éles telepítés**. Ugyanígy: MinIO nélkül nincs Velero, Loki és Harbor-tárolás; Keycloak nélkül nincs SSO a Grafana/ArgoCD/Rancher mögé; PostgreSQL nélkül nincs Keycloak. A sorrend megfordítása ideiglenes kézi megoldásokhoz vezet, amelyek később csak jelentős ráfordítással válthatók ki.

4. szint A szállítási lánc (CI→CD) összezárása

Elem	Előfeltétel	Mit tesz lehetővé
External Secrets Operator	Vault + RKE2	titkok automatikus K8s-be juttatását
ArgoCD + konfigurációs repo + app-of-apps	RKE2 + Git (+SSO-hoz Keycloak; éles secretekhez ESO)	a GitOps-telepítést — innentől élesítés = merge
Céges Helm library chart + alkalmazás-chartok	ArgoCD + Harbor	egységes, életjel-végpontokkal ellátott telepítéseket
CI biztonsági szkennerek (Trivy, Gitleaks, Semgrep, Hadolint, Syft, Cosign) és függőség-frissítés (Renovate)	CI + Harbor	az ellátásilánc-kontrollokat, audit-bizonyítékot
Saját (on-prem) CI runnerek a klaszteren	RKE2 + CI	az ingyenes keret utáni build-kapacitást
Nyelvi pipeline-ok véglegesítése (Java, Go, Node.js, PHP…)	CI + szkennerek + Harbor	a fejlesztőcsapatok teljes átállását
Velero (ütemezett mentés MinIO-ba)	RKE2 + MinIO	a Kubernetes-szintű mentést
pgBackRest (PostgreSQL PITR) + XtraBackup (MySQL)	adatbázisok + MinIO	az adatbázis-mentést

Release stratégiák a meglévő eszközökkel: rolling finomhangolás; blue-green/canary Helm+ArgoCD+ingress útvonalon; feature flag konfigurációból	ArgoCD + Helm + ingress	a kockázathoz igazított élesítést
Séma-migrációs fegyelem (expand-contract, nyelvi natív eszközök, migráció külön Jobként)	CI + adatbázisok + ArgoCD	a leállás nélküli adatbázis-változtatásokat

FOGALOM

RPO és RTO — a mentési stratégia két kulcsszáma. Az RPO (Recovery Point Objective): legfeljebb mennyi adat veszhet el — ezt a mentések gyakorisága határozza meg. Az RTO (Recovery Time Objective): legfeljebb mennyi idő alatt kell talpra állni — ezt a visszaállítás begyakorlottsága. A nem tesztelt mentés csak remény: az RPO/RTO számok negyedéves, dokumentált visszaállítási próbával válnak ténnyé.

5. szint Láthatóság és mérés

Elem	Előfeltétel	Mit tesz lehetővé
Prometheus + Grafana + Alertmanager (exporterekkel)	RKE2 (+Keycloak SSO)	metrikákat és riasztást
Loki + Alloy naplógyűjtés	RKE2 + MinIO	központi naplózást
Tempo + OpenTelemetry Collector	RKE2 + MinIO (+instrumentálás)	hívásláncokat, korrelációt
Sentry (külön VM) + Grafana Faro	IaC (VM) / Alloy	hibakövetést és frontend-mérést
RabbitMQ (és Kafka, ha a volumen indokolja)	RKE2	aszinkron feldolgozást, puffert
ITOps (mlops.hu) — SLO/SLA mérés, SLA-portál, automatikus ticket *(opcionális)*	RKE2 + futó szolgáltatások + Prometheus	az SLA-riportokat, error budgetet, státuszoldalt
Skálázás és kapacitástervezés: HPA + PodDisruptionBudget, node-bővítés IaC-ből, kifogyás-riasztások, negyedéves kapacitás-review	RKE2 + Prometheus + IaC	hogy a terhelés-növekedés ne incidens legyen
Terheléstesztelés (k6, staging környezet ellen)	staging környezet + mérési stack	az erőforrás- és SLO-beállítások megalapozását
Költség-láthatóság: igényelt vs. tényleges fogyasztás dashboardok, rightsizing	Prometheus + Grafana	a vezetői költség-riport adatait, felszabadítható kapacitást

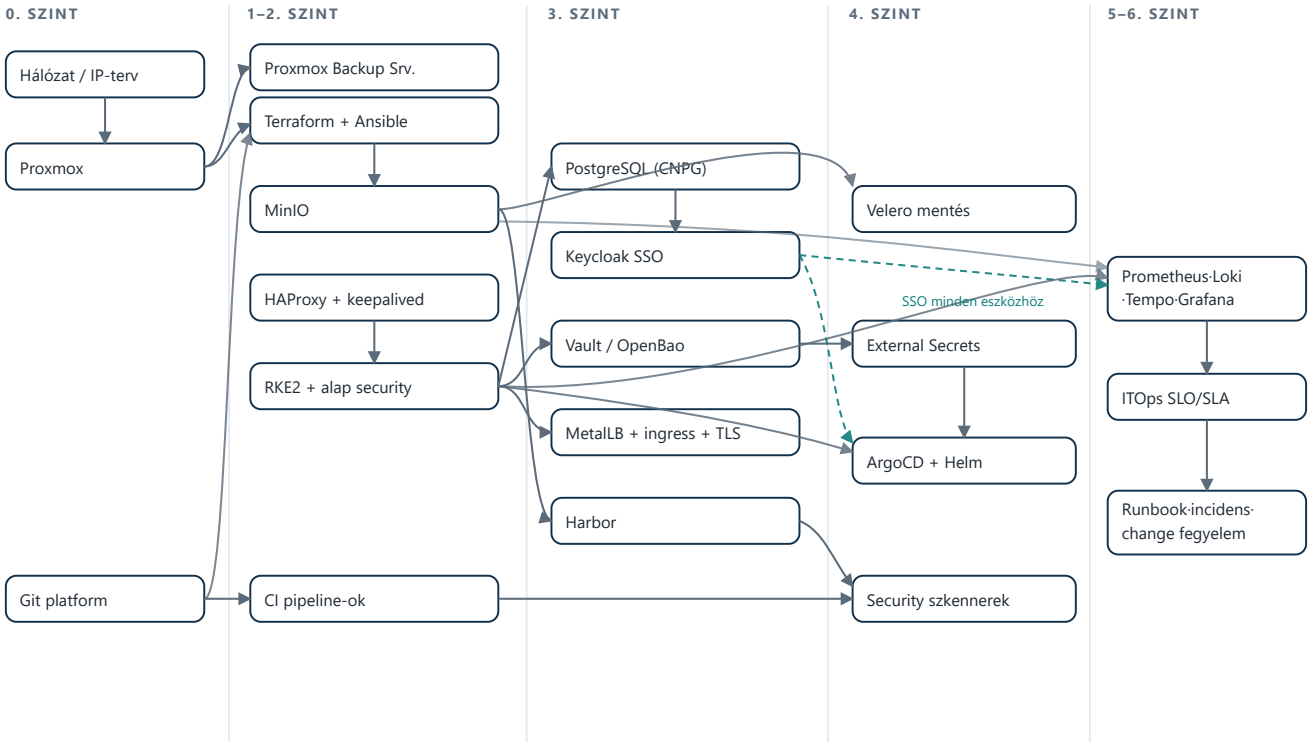
FOGALOM

Error budget (hibaköltségvetés) — ha az SLO 99,9%, akkor havonta ~43 perc „megengedett” kiesés van. Ez nem kudarc-keret, hanem kockázatvállalási valuta: amíg van belőle, lehet bátran változtatni; ha elfogyott, a stabilizálás előzi a fejlesztést. A vezetőség és a mérnökök közti vitákat számmá alakítja.

6. szint Fegyelem és érettség — a komplex vég

Elem	Előfeltétel	Mit tesz lehetővé
Runbookok minden riasztáshoz és szolgáltatáshoz, ügyeleti rend, incidensfolyamat	Alertmanager + Confluence	a kiszámítható üzemeltetést
Változáskezelés (a PR mint change record, freeze-ablakok)	GitOps-lánc	az auditor-kompatibilis változáskezelést
DR-tesztek (negyedéves visszaállítási próba), hozzáférés-felülvizsgálat	mentési stack + Keycloak	az audit-bizonyítékokat
NIS2 / ISO 27001 megfelelés-térkép karbantartása	a fenti kontrollok	a megfelelőséget mint melléktermeket
L7 védelem finomhangolása: rate limiting a meglévő HAProxy/Traefik eszközökkel, admin-felületek VPN mögé	HAProxy + Traefik + VPN	az alkalmazás-szintű visszaélések elleni védelmet
Üzletmenet-folytonosság: üzleti hatáselemzés, degradált működési tervek, éves gyakorlat	mentési stack + vezetőség	hogy IT-kiesés alatt a szervezet működőképes maradjon
[Opcionális] Kyverno → Falco → Trivy Operator (ebben a sorrendben)	stabil K8s + érett csapat	a Kubernetes-biztonság következő szintjét
[Opcionális] Istio service mesh (csak indokolt esetben), Thanos/Mimir	mesh-igény / retention-igény / érett CI	a nagyvállalati érettséget

A függőségi gráf — a gerinc



5. ábra — A függőségi gerinc: minden nyíl egy „ennek készen kell lennie előbb” viszony (szaggatott: az SSO-bekötés, ami minden platformelemet érint)

Gyakori hibák

- **Szint-alapú „vízesés”:** megvárni egy teljes szint végét, pedig csak az adott elem saját előfeltétele számít.
- **A mentés halogatása** az 5–6. szintig — a Proxmox Backup Server már az 1. szinten bevezethető, és kell is.
- **Eszköz bevezetése SSO nélkül** — a kilépő munkatárs hozzáférése az adott eszközben nem vonható vissza központilag.
- **Az opcionális elemek előrehozása** (Kyverno, Falco, Istio), mielőtt az alapok stabilak.
- **A fejlesztő-edukáció kihagyása:** a 4. szint (nyelvi pipeline-ok, életjel-végpontok) csak a fejlesztőkkel együtt megy át.

4. FEJEZET

Kommunikáció a vezetőség felé

> *A vezetőség három dolgot ért: kockázatot, pénzt és időt. Ez a fejezet arról szól, hogyan fordítsuk le a DevOps-működést erre a három nyelvre — mérőszámokkal, egyoldalas havi riporttal, és azzal is, mit mondjunk, amikor éppen áll a rendszer.*

Alapszabály: üzleti nyelv

A vezetőség három dimenzióban gondolkodik: **kockázat, pénz, idő**. Minden technikai üzenetet ebbe a három dimenzióba kell lefordítani, mielőtt vezetői szintre kerül.

Technikai állítás	Üzleti fordítás
„Az etcd nincs titkosítva"	„Ha ellopják a szerver diszkjét vagy a mentést, minden jelszavunk és ügyféladatunk olvasható — ez GDPR-bírság és a NIS2 szerinti meg nem felelés kockázata."
„Nincs staging környezet"	„Minden változtatást élesben tesztelünk — minden release az ügyfelek előtt bukhat el; egy hibás telepítés X óra kiesés."
„A MySQL-nek nincs replikája"	„Ha az az egy szerver meghal, a rendelési rendszer áll, amíg mentésből vissza nem állunk — mérésünk szerint 2–4 óra, óránként Y Ft kieső bevétel."
„Bevezetjük az ArgoCD-t"	„Minden változás visszakövethető és 10 perc alatt visszavonható lesz — csökken a hibás release-ek helyreállítási ideje és az audit-felkészülés költsége."

Gyakorlati szabályok:

- **Következménnyel kezdeni, nem technológiával.** Nem „Kyverno policy engine kell", hanem „jelenleg bárki futtathat root konténert a klaszterben — ez a kockázat, ennyibe kerül megszüntetni".
- **Számszerűsíteni.** „Lassú a release" helyett: „egy változtatás ma átlag 9 nap alatt jut élesbe, a cél 1 nap". Ha nincs pontos szám, becslés intervallummal — a becsült szám is jobb, mint a jelző.
- **Opciókat adni, nem ultimátumot.** A vezetőség dönteni akar, nem jóváhagyni. Minden kéréshez legalább két opció + a „nem csinálunk semmit" opció következménye.
- **A rendszeres riport rövid, az eseti mélyanyag külön.** A havi riport 1 oldal; ha a vezető többet kér, mellékletben van a részlet.

Mit mérünk és mutatunk

DORA metrikák magyarul

A DORA (DevOps Research and Assessment, a Google DevOps-kutatási programja) négy kulcsmetrikája a szoftverszállítás sebességét és stabilitását méri — vezetőség felé ez a legjobban bevált, iparági viszonyítási

számokkal alátámasztható keret. A DORA-kutatás visszatérően igazolt megállapítása, hogy a „gyorsan szállítunk” és a „nem törünk el semmit” *nem* egymás rovására megy: a legjobb csapatok mindkettőben jók.

6. ábra — A négy DORA-metrika és az „elit” szint iparági viszonyítási értékei

Metrika	Mit mond a vezetőnek
Deployment frequency — telepítési gyakoriság	Milyen gyorsan jut el egy fejlesztés a piacra, az ügyfélhez.
Lead time for changes — átfutási idő	Mennyi idő alatt reagálunk üzleti igényre vagy hibára.
Change failure rate — hibás változtatások aránya	Mennyire kockázatos nálunk egy release.
MTTR (time to restore) — helyreállítási idő	Ha baj van, mennyi ideig érinti az ügyfeleket.

Vezetői tálaláshoz: a négy szám **trendben** (elmúlt 3–6 hónap) többet mond, mint az abszolút érték. A DORA-kutatás szerint az elit és a leggyengébb csapatok között nagyságrendi — több százszoros — különbség van a telepítési gyakoriságban és a helyreállítási időben; ez jól használható érv arra, hogy a DevOps-beruházás mérhető üzleti előnyt ad, nem belső kényelmi fejlesztés.

A DORA mellett mutatandó számok

- **Rendelkezésre állás (SLA/SLO-teljesülés)** — a platformban az ITOps adja a mérést és a havi riportot: szolgáltatásonként hány % volt az üzemidő, hol tart a hibaköltségvetés.
- **Incidensek** — darabszám súlyosság szerint, ügyfelet érintő percek, fő ok-kategóriák. Nem a nyers riasztásszám (az üzemeltetési belügy), hanem az üzleti hatással járó események.
- **Biztonsági helyzet** — nyitott kritikus/magas sérülékenységek száma és trendje, javítási átfutási idő, hozzáférés-felülvizsgálat státusza, audit-készültség. Számok és trend, nem CVE-lista.
- **Költség** — az infrastruktúra havi költsége (felhőszámla vagy on-prem amortizáció + energia + licenc), trend, és a következő ismert költség-esemény.

Havi riport — egyoldalas sablon

A riport ritmusa: **havi, fix időpontban, mindig ugyanabban a szerkezetben.** A vezető 2 perc alatt átfutja; ami zöld, arról nem kell beszélni.

IT/DevOps havi riport — minta

Terület	Státusz	Megjegyzés (csak ha nem zöld)
Rendelkezésre állás	🟢 99,96%	SLO: 99,9% — teljesül
Release-folyamat	🟢	34 telepítés, 1 visszavonás
Biztonság	🟡	2 kritikus sérülékenység javítása folyamatban
Compliance / audit	🟢	NIS2-akcióterv ütemben
Költség	🟡	+8% (új staging környezet — tervezett)

Kulcsszámok (trend az előző 3 hónaphoz képest): telepítés 34/hó (↑) · átfutási idő 1,5 nap (↓, cél: 1 nap) · hibás változtatás 3% (→) · helyreállítás 25 perc (↓) · 1 db SEV2 incidens (40 perc, becsült hatás ~X Ft).

Top 3 kockázat: 1) A MySQL egyetlen szerveren fut — kiesésnél 2–4 óra állás; javasolt: replika. 2) A titoktár gyártói támogatás nélkül fut — audit-találat várható. 3) Az ügyelet 1 főn áll — szabadság idején nincs lefedettség.

Döntést kérünk: ☐ MySQL-replika beszerzése (~X Ft egyszeri; 1. kockázat) — határidő: szept. 15. ☐ Ügyeleti pótlék rendezése a 2. fő bevonásához (havi ~Y Ft).

Szabályok a sablonhoz:

- **Jelzőlámpa-fegyelem:** a sárga/piros mindig kap egy sor magyarázatot és egy tervezett lépést. Zöldhöz nem írunk szöveget. Ha minden hónapban minden zöld, a riport hiteltelen — a kockázat-szekció akkor is éljen.
- **Top 3 kockázat:** mindig pontosan három, rangsorolva. Ez kényszeríti ki a priorizálást, és a vezetőség hozzászokik, hogy itt kell figyelni.
- **Kért döntések:** konkrét, határidős, összeggel. Ha nincs döntési igény, a szekció tartalma „nincs” — de a szekció maga mindig ott van.

Költség-kommunikáció

On-prem vagy felhő — TCO-érvelés

A „felhő vagy saját vas” kérdésben a vezetőség felé teljes birtoklási költséggel (TCO) kell érvelni, nem listaárral. On-prem oldalon: hardver-amortizáció (3–5 év), áram és hűtés, hely, hálózat, támogatási szerződések — és a **munkaidő**. Felhő oldalon: a havi számla mellett a kilépési költség (adatkiviteli díjak, gyártófüggés) és az árváltozási kitettség. Vezetői szinten is használható irányelvek:

- Kiszámítható, egyenletes terhelésnél az on-prem 2–4 év távon jellemzően olcsóbb; erősen ingadozó vagy gyorsan növekvő terhelésnél a felhő rugalmassága ér többet.
- A hibrid nem kudarc, hanem stratégia: állandó terhelés on-prem, csúcs és katasztrófa-tartalék felhőben.
- A számítást évente frissíteni kell — a TCO-tábla a havi riport melléklete lehet, amikor releváns döntés van napirenden.

> **FOGALOM**
>
> **TCO** — Total Cost of Ownership, teljes birtoklási költség: nem a beszerzési ár, hanem minden költség a teljes életciklusra — beszerzés, üzemeltetés, energia, licenc, munkaidő, csere. Két megoldás csak TCO-alapon hasonlítható össze tisztességesen.

Licencköltségek előrejelzése

A platform alapelve az ingyenes, nyílt forráskódú eszköz — de a vezetőség felé előre kell jelezni, **mikor és miért** merülhet fel mégis licencköltség: (1) egy audit vagy megfelelőségi követelmény gyártói támogatást ír elő; (2) egy nyílt projekt licencet vált vagy az ingyenes szint szűkül — erre volt már iparági precedens, ezért a kritikus elemek mellé mindig van nyílt alternatíva betervezve; (3) felhasználószám-alapú szintlépés a Git-platformnál. A költségvetés-tervezéshez éves licenc-előrejelzési tábla készüljön: eszköz / jelenlegi költség / ismert kockázat / várható változás dátuma.

„Miért kell 3 gép, ha 1 is elég?" — redundancia üzleti nyelven

Ez a leggyakoribb vezetői visszakérdezés, és technikai válasz helyett üzleti választ igényel:

- **A 3 gép nem 3× kapacitás, hanem folyamatos működés.** Egy gépnél minden karbantartás és minden meghibásodás leállás. Háromnál egy gép kiesését az ügyfél észre sem veszi.
- **Számokkal:** 1 gép esetén évi X óra tervezett karbantartási leállás + meghibásodási kockázat; az óránkénti kiesési költséggel szorozva a két plusz gép ára jellemzően egyetlen komolyabb incidens alatt megtérül.
- **A páratlan szám műszaki követelmény:** két gépnél a rendszer meghibásodás esetén nem tud dönteni, három gépnél többségi döntés születik.
- **A redundancia biztosításként értelmezhető:** a költsége a kiesések elkerülésében, a folyamatos működésben térül meg.

Beruházás-kérés felépítése

Minden vezetői jóváhagyást igénylő kérés ugyanazon a négylépcsős íven halad:

1. **Probléma** — tényszerűen, egy mondatban. („A mentéseket soha nem teszteltük visszaállítással.")
2. **Kockázat forintosítva** — mi történik, ha nem lépünk, mekkora valószínűséggel, mennyibe kerül. („Ha az adatbázis sérül és a mentés nem áll vissza, a teljes rendelési történet elveszhet; egy nap kiesés ~X Ft bevétel + helyreállítási munka + hírnévkár. Iparági tapasztalat, hogy a nem tesztelt mentések jelentős része élesben nem visszaállítható.")
3. **Opciók** — legalább kettő + a nulla-opció, mindegyik költséggel és maradék kockázattal: A) nem csinálunk semmit — 0 Ft, a kockázat marad; B) negyedéves kézi visszaállítási próba — ~2 embernap/negyedév; C) automatizált próba-visszaállítás riporttal — egyszeri X embernap, utána minimális.
4. **Javaslat** — melyiket javasoljuk, miért, mikorra, ki csinálja. A döntés a vezetőé, a javaslat a miénk — a javaslat nélküli opciólista döntésképtelenséget sugall.

Incidens-kommunikáció vezetőknek

Súlyos, ügyfeleket érintő incidens (SEV1) esetén a vezetőség tájékoztatása nem udvariasság, hanem a folyamat része — de más csatornán és más nyelven, mint a technikai hibaelhárítás.

- **Mikor:** első értesítés az incidens visszaigazolása után 15–30 percen belül — akkor is, ha még nincs diagnózis. A vezetőség az incidensről ne külső forrásból értesüljön.
- **Ki:** az incidens-parancsnok vagy kijelölt kommunikációs felelős — soha nem a hibát éppen elhárító mérnök.
- **Mit:** üzleti hatás, nem technikai ok. Sablon az első üzenetre:

```
[SEV1] Webshop nem elérhető – 14:05 óta
Hatás: ügyfelek nem tudnak rendelni; belső rendszerek működnek.
Állapot: hibaelhárítás folyamatban, az ok azonosítása zajlik.
Következő tájékoztatás: 15:00-kor, vagy előbb, ha érdemi változás van.
Kapcsolat: [név, telefon]
```

- **Ritmus:** előre bejelentett időközönként (pl. óránként) frissítés akkor is, ha nincs új információ. A megígért és betartott ritmus megszünteti a „mi újság?” hívásokat, amelyek a hibaelhárítást lassítják.
- **Ne:** találgatott ok („valószínűleg a hálózat”), találgatott határidő („10 perc és kész”), bűnbak. Az ok-elemzés az utólagos kiértékelés dolga — annak eredményét (vádaskodásmentes összefoglaló + intézkedések) a vezetőség is megkapja.

> FOGALOM
>
> **Blameless postmortem** — incidens utáni kiértékelés, amely a rendszer- és folyamathibákat keresi, nem a bűnöst. Nem jóindulat kérdése, hanem hatékonyságé: ahol a hibázót szankcionálják, ott a hibákat eltitkolják, és a hibaokok ismétlődnek.

Audit- és megfelelőségi vonatkozás

A vezetői riportok és incidens-értesítések megőrzendő bizonyítékok: a NIS2 szerinti incidensbejelentési kötelezettség (korai figyelmeztetés 24 órán belül a hatóság felé jelentős incidensnél) vezetői döntést igényel — a belső eszkalációs láncnak ezért dokumentáltnak kell lennie. A havi riportok sorozata auditnál azt bizonyítja, hogy a vezetőség rendszeresen tájékozódott a kockázatokról (ISO 27001: vezetői átvizsgálás).

Gyakori hibák

- **Technikai zsargon a vezetői anyagban** — a „pod eviction miatt OOMKilled” a vezetőnek nem hordoz információt; a fordítás a technikai csapat feladata.
- **Riasztás-özön a vezetőnek** — a vezető nem ügyeletes; ha minden riasztást megkap, a valódi vészhelyzetet sem veszi észre. Vezető csak SEV1/SEV2 összefoglalót kap, szűrten.
- **Csak rossz hírek** — ha a DevOps csak akkor jelentkezik, amikor pénz kell vagy baj van, költséghelynek látszik. A havi riport a sikereket is mutatja.

- **Számok kontextus nélkül** — a „99,5% üzemidő” jól hangzik, pedig havi ~3,6 óra kiesés; mindig cél mellé állítva és időtartamra lefordítva.
- **Döntési kérés opciók és összeg nélkül** — a „kellene egy erősebb szerver” nem döntéselőkészítés.
- **A riport-ritmus felborítása** — kihagyott hónapok után a riport újraindítása bizalomvesztésből indul.
- **Ígért határidő incidens közben** — csak azt ígérjük, amit kontrollálunk: a következő tájékoztatás időpontját.

A fejezet forrásai

1. DORA / Google Cloud — Use Four Keys metrics to measure your DevOps performance
2. DORA 2025 State of DevOps riport és összefoglalói (DevOps.com, Faros.ai, Splunk)
3. Octopus Deploy — The 4 DORA metrics and top findings from the 2024/25 DORA report

UTÓSZÓ

Amin a bevezetés valójában múlik

A lista a végére ért, de a munka nem: a bevezetés sorsa ritkán a technológián múlik. A forgatókönyv bármely szervezetre ráilleszthető — ami ezután következik, az három dolgon dől el.

Az első a fegyelem. Egyetlen „csak most az egyszer” kézi beavatkozás elég ahhoz, hogy a Git és a valóság elváljon egymástól — és onnantól a rendszer legértékesebb tulajdonsága, a kiszámíthatóság vész el. A break-glass (vészhelyzeti kézi beavatkozási) eljárás nem megengedi, hanem kezelhetővé teszi a kivételt: dokumentálva, utólag mindig a kódba visszavezetve.

A második a fejlesztői kompetencia. A platform annyit ér, amennyit a fejlesztők használni tudnak belőle. Az edukációra fordított idő nem költség, hanem a leggyorsabban megtérülő befektetés.

A harmadik a kommunikáció fenntartása — a leginkább alábecsült tényező. A technikai csapat hajlamos azt hinni, hogy a jól működő rendszer önmagáért beszél. Nem beszél: a költségvetési döntések asztalánál csak az létezik, amiről riport készül. A platform az ehhez szükséges számokat magától termeli; a 4. fejezet ritmusa arról gondoskodik, hogy el is jussanak oda, ahol döntenek róluk.

A folytatás irányát a függelék jelöli ki: huszonöt terület, amelyek kidolgozásával a forgatókönyvből működő rendszer lesz — telepítési mintákig és megfelelési térképekig lebontva.

FÜGGELÉK

Mélységében kidolgozandó témakörök

Ez a könyv forgatókönyv, nem kézikönyv: megmutatja, mi mire épül, de az egyes elemek mélységi kidolgozása — telepítési minták, konfigurációk, döntési szempontok, audit-vonatkozások — önálló dokumentációt igényel. Az alábbi lista azt rögzíti, mely területekről van szó; a számozás a javasolt tematikai bontást követi.

1. **[1] Verziókezelés és repo-szabályzat** — GitHub és GitLab governance, branch protection és rulesets, platformválasztási döntési mátrix költségszámokkal, migráció mindkét irányba, branching-stratégiák (trunk-based, GitHub Flow, GitFlow).
2. **[2] CI/CD pipeline-ok** — GitHub Actions az ingyenes kerettől az on-prem runner-flottáig, GitLab CI runnerek és executorok, a teljes Docker build lánc (multi-stage, cache, multi-arch), biztonsági szkennerek pipeline-ba illesztése fail-kritériumokkal.
3. **[3] Nyelvspecifikus CI/CD kézikönyvek** — Java, Go, Node.js, PHP, Python és .NET: verziópolitikák, build- és teszteszközök, konténerizálási minták, teljes minta-pipeline-ok mindkét CI-rendszerre, Kubernetes-sajátosságok nyelvenként.
4. **[4] Container registry és tárolás** — GHCR/ECR/Harbor döntés valós költségadatokkal, Harbor-üzemeltetés (szkennelés, replikáció, proxy cache), MinIO-architektúra, objektumzárolás és a tárolási réteg kockázatelemzése alternatívákkal.
5. **[5] GitOps mélységben** — repo-struktúrák és környezetkezelés, ArgoCD magas rendelkezésre állással, szinkronizálási stratégiák és hullámok, Helm library chartok és chart-tesztelés, release stratégiák (rolling, blue-green, canary, feature flag) a meglévő eszközökkel, katasztrófa utáni újraépítés Gitből.
6. **[6] Infrastructure as Code** — Terraform/OpenTofu és Terragrunt (state-kezelés, modulok, CI-integráció), Proxmox-provisioning cloud-init sablonokkal, Ansible-szerepek és dinamikus inventory, licenchelyzet-elemzés.
7. **[7] Identitás és titokkezelés** — Vault/OpenBao telepítési és auth-minták, dinamikus adatbázis-hitelesítők, External Secrets, Keycloak SSO-bekötés eszközönként (kilenc integráció), teljes belépő-kilépő folyamat checklistákkal.
8. **[8] Kubernetes-üzemeltetés** — disztribúció-választás (RKE2/k3s/vanilla) és a „military grade” tartalma, HA-telepítés és frissítési folyamat, terheléselosztás és ingress a teljes forgalmi úttal, Istio döntési keret, a K8s-biztonság rétegei alapszinttől a runtime-védelemig, skálázás és kapacitástervezés (HPA, node-bővítés, terheléstesztelés).
9. **[9] Megfigyelhetőség** — Prometheus/Grafana/Loki/Tempo telepítési és méretezési minták, OpenTelemetry-instrumentálás, Sentry és Faro, Kafka/RabbitMQ döntési keret, az SLO/SLA-mérés és -riportolás teljes felépítése, valamint a költség-láthatóság a meglévő metrikákból.
10. **[10] Adatbázis-üzemeltetés** — PostgreSQL Kubernetesen és VM-en (operátor, PITR-mentés, kapcsolat-pooling), MySQL bare metalon (replikáció, fizikai mentés), Redis/Valkey topológiák és licenchelyzet, séma-migráció fegyelme (expand-contract, online DDL).
11. **[11] Biztonság és megfelelőség** — vállalati kiberbiztonsági program, NIS2 és a magyar kiberbiztonsági törvény gyakorlati megfelelés-térképe határidőkkel, ISO 27001 / SOC 2 / DORA / GDPR DevOps-fordításban, ellátásilánc-biztonság (SBOM, aláírás, SLSA), audit-bizonyítékok és felkészülési checklist.
12. **[12] Üzemeltetési fegyelem** — runbook-sablonok és tudásbázis-struktúra, incidenskezelés és ügyeleti rend, változáskezelés GitOps-korban, a teljes mentési és katasztrófa-helyreállítási architektúra RPO/RTO-táblákkal,

valamint az üzletmenet-folytonosság (üzleti hatáselemzés, degradált működés, krízisstáb).

13. **[13] Hálózat és a további kommunikációs irányok** — hálózati alapok DevOps-szemmel és a netops-együttműködés sablonjai, DNS/TLS/tanúsítvány-kezelés, a Kubernetes körüli hálózat, L7 védelem a meglévő eszközökkel (rate limiting, admin-felületek zárása); kommunikáció a fejlesztők felé (életjel-végpontok, edukációs program) és a technikusok felé (eszkalációs mátrix, átadás-átvétel), szerepkörök és RACI.

A tématérkép újabb területei

A fenti tizenhárom terület a platform gerincéhez tartozik. Az 1. fejezetben bemutatott, öt forrás keresztezéséből készült tématérkép ezeken túl további területeket azonosított — jellemzően olyanokat, amelyeket az eszközközpontú tervezés hagy ki, pedig audit- vagy üzemeltetési szempontból előbb-utóbb megkerülhetetlenek:

1. **[14] Túlterhelés-kezelési minták** — retry és backoff, circuit breaker, load shedding, a kaszkádhibák megelőzése.
2. **[15] Problem management** — a visszatérő hibák és az ismert workaroundok strukturált nyilvántartása, a postmortemeken túl.
3. **[16] Production readiness review** — formális éles-előtti átvilágítás, go-live forgatókönyv, megerősített ügyeleti (hypercare) időszak.
4. **[17] Ütemezett feladatok üzemeltetése** — CronJob-fegyelem, kihagyott futások észlelése, idempotencia.
5. **[18] Alap-infrastruktúra higiénia** — óraszinkron (NTP), e-mail hitelesítés (SPF, DKIM, DMARC), kvóta- és limit-nyilvántartás.
6. **[19] Végponti védelem és HR-biztonság** — a munkaállomások védelme, a belépés-kilépés biztonsági lépései, security awareness program.
7. **[20] Threat modeling és hibamód-elemzés** — a kockázatok szisztematikus feltérképezése még tervezéskor.
8. **[21] Incidens-forenzika** — bizonyíték-megőrzés, chain of custody, hatósági együttműködés rendje.
9. **[22] Fizikai biztonság és lokáció** — szerverterem-minimumok, colocation-döntés, a második telephely megválasztása.
10. **[23] Domain-szintű redundancia** — DNS-alapú forgalom-átterelés telephelyek között, alacsony TTL-stratégia.
11. **[24] Adatfeldolgozó (batch/ETL) pipeline-ok** — késés-riasztás, újrafuttathatóság, adatminőség mint üzemeltetett szolgáltatás.
12. **[25] Vegyes szerverpark** — a Windows-gépek bevonása a monitoringba, a mentésbe és az automatizálásba.

DevOps bevezetés a gyakorlatban

Egy DevOps-bevezetés sorsát ritkán az eszközválasztás dönti el; sokkal inkább az, hogy a jó elemek milyen sorrendben és milyen fegyelemmel állnak össze rendszerré. Ez a könyv erre a sorrendre ad forgatókönyvet, egyetlen szabály köré építve: egy elem akkor kezdhető el, amikor minden előfeltétele kész. A hét szint, az ábrák és a táblázatok ezt a szabályt bontják ki az atomi elemektől a komplex, nyílt forráskódú platformig.

Külön fejezet szól arról, ami a technikai anyagokból rendre kimarad: hogyan képviselhető mindez a vezetőség felé — mérőszámokkal, egyoldalas riporttal és döntésre kész javaslatokkal.

Puskás Balázs · DevOps
Mobil: +36 30 680 0800
E-mail: puskas.balazs@36306800800.hu