

Dokploy UI – Új site létrehozása és űrlap kitöltése

Új site űrlap mezői és tipikus beállításai

Az új site létrehozásakor a Dokploy UI egy űrlapot kínál fel, amelyben meg kell adni néhány kulcsfontosságú paramétert. Az alábbiakban sorra vesszük a mezőket és ismertetjük, hogy mit jelentenek, illetve milyen értékeket érdemes beállítani tipikus esetben:

- **Host (Domain név):** Itt kell megadni azt a *domain nevet*, amely alatt az alkalmazást elérhetővé szeretnénk tenni. Példa: `app.sajatdomain.hu` vagy `api.sajatdomain.hu`. Ügyeljünk arra, hogy a domain név DNS-beállításai megfelelőek legyenek, vagyis mutassanak arra a szerverre, ahol az alkalmazás fut ¹. (A DNS konfiguráció részleteire később kitérünk.) Ha nem rendelkezünk saját domainnel, a Dokploy lehetővé teszi egy ingyenes *traefik.me* domain generálását is (egy „dobókocka” ikon segítségével), ami egy olyan *“varázs” domain*, amely automatikusan a mi szerverünk IP-címére irányít minden kérést ². Ez utóbbival gyorsan beállíthatunk egy teszt domaint a fejlesztéshez konfiguráció nélkül, azonban ez alapértelmezetten csak HTTP-n (titkosítatlanul) fog működni ³.
- **Path (Útvonal):** A domainen belüli útvonal, amin az alkalmazás elérhető. Alapértelmezésben ez általában `/` (a gyökér útvonal), ami azt jelenti, hogy a megadott domain alatt közvetlenül érhető el az alkalmazás. Beállíthatunk konkrét alútvonalat is, például `/app` vagy `/api`, ha azt szeretnénk, hogy az alkalmazás ne a gyökéren, hanem egy aloldalon legyen elérhető (pl. `sajatdomain.hu/app`) ⁴. Fontos tudni, hogy ha nem a gyökér útvonalat használjuk, akkor az alkalmazásnak is tudnia kell kezelni ezt a prefixet, vagy a Traefik proxy beállításain keresztül lehet ezt módosítani (lásd **Strip Path** és **Internal Path**).
- **Internal Path (Belső útvonal):** Opcionális beállítás, amely megadja, hogy a **konténerben futó alkalmazás milyen útvonalat vár** a kérések elején. Alap esetben az alkalmazások a gyökérrel kezdődő kéréseket várják (pl. `GET /valami`), így az *Internal Path* értékét ilyenkor üresen hagyjuk vagy `/`-re állítjuk (nincs belső prefix). Ha azonban az alkalmazásunk úgy van konfigurálva, hogy minden kérést egy bizonyos prefixszel vár (például a backend szolgáltatás egy `/api` alapon működik), akkor itt megadhatjuk ezt a prefixet. A Traefik ilyenkor a bejövő kérések elé odateszi ezt a belső útvonalat, mielőtt továbbítaná a konténer felé ⁵ ⁶. **Példa:** Ha az *Internal Path* `/backend/api`, a fenti beállítással egy `api.sajatdomain.hu/v1/users` kérés a konténer felé `/backend/api/v1/users` formában kerül továbbításra ⁷. Ezt a mezőt csak akkor szükséges kitölteni, ha az alkalmazás kifejezetten igényli; tipikus webalkalmazásoknál (pl. Node.js/Express app) erre általában nincs szükség, hagyjuk üresen.
- **Strip Path:** Ez egy kapcsoló/opció, amellyel beállítható, hogy a megadott *Path* (külső útvonal prefix) **levágásra kerüljön** a HTTP kérésből, mielőtt az eljut az alkalmazáshoz. Ennek akkor van jelentősége, ha a nyilvános URL-ben használunk valamiféle prefixet, de azt szeretnénk, hogy a konténerben futó alkalmazás *erről ne is tudjon*, azaz „tisztán” (prefix nélkül) kapja meg az útvonalat. **Mikor engedélyezzük?** Ha például a *Path* mezőben beállítottuk `/dashboard`-ra az alkalmazást, de maga az alkalmazás a kéréseket a gyökéren várja, akkor érdemes a *Strip Path*-et

bekapcsolni. Így amikor a felhasználó a `app.sajatdomain.hu/dashboard/oldal` URL-t éri el, a Traefik levágja a `/dashboard` előtagot, és a konténer felé már csak `/oldal` kérést továbbít ⁸. **Egyszerűen fogalmazva:** használd a *Strip Path*-et, ha **a külső útvonalat nem akarod megmutatni az alkalmazásnak**. Arra is van lehetőség, hogy a *Strip Path*-et és az *Internal Path*-et egyszerre alkalmazd összetettebb esetekben – ilyenkor először a megadott prefix levágásra kerül, majd egy másik prefix (Internal Path) hozzáadódik a kéréshez ⁹ ¹⁰. Érdemes ezeket a funkciókat óvatosan használni és alaposan tesztelni, mert ha az alkalmazás például abszolút URL-eket generál a válaszokban a saját maga által ismert útvonalak alapján, könnyen vezethetnek **átírányítási hibákhoz** vagy nem várt működéshez, ha a path-ok nincsenek szinkronban ¹¹.

- **Container Port (Konténer port):** A konténeren **belüli** portszám, amin az alkalmazás hallgat. Ide annak a portnak a számát írjuk, amelyet az adott alkalmazás tipikusan használ. **Például:** egy Node.js alapú webalkalmazás (pl. Express vagy Next.js) alapértelmezésben a `3000`-es porton fut a konténeren belül, így ebbe a mezőbe `3000` kerül ¹². Más gyakori értékek: egy Astro keretrendszerű alkalmazás pl. `4321` porton fut fejlesztői módban ¹³; egy statikus fájlokat kiszolgáló Nginx vagy Apache konténer pedig jellemzően a `80`-as portot használja belsőleg. (Fontos megjegyzés: ha a Dokploy-ban *Static* típusú buildet vagy Nixpacks *Publish Directory* beállítást használunk egy alkalmazásnál, akkor **a domain létrehozásakor a konténer port mezőbe `80`-at kell megadni**, mivel az így épített konténer a 80-as porton szolgálja ki a tartalmat ¹⁴.) **Ne feledd:** a *Container Port* mező **nem nyit meg** nyilvános portot a szerveren, csak azt határozza meg, hogy a Traefik proxy mely konténer-port felé irányítsa a forgalmat ¹⁵. A Traefik a háttérben, belső Docker hálózaton továbbítja a kéréseket erre a konténerportra. (A Dokploy-ban létezik egy külön "*Advanced -> Ports*" szekció is, ahol közvetlenül ki lehetne nyitni portokat a külvilág felé – de ha domainen keresztül, Traefikkal érjük el az alkalmazást, erre általában nincs szükség.)

- **HTTPS kapcsoló:** Ezzel a kapcsolóval állíthatjuk be, hogy a domain **HTTPS alatt (titkosítva)** működjön-e. Ha *HTTPS = ON*, akkor a Dokploy/Traefik automatikusan intézi, hogy a domainre beérkező kérések TLS titkosítással legyenek kiszolgálva. **Mikor célszerű bekapcsolni?** Szinte *mindig, amikor lehetséges*. **Ajánlott** éles, publikus alkalmazásoknál a HTTPS használata a biztonság és a bizalom érdekében ¹. Amennyiben saját, vásárolt domain nevet használunk, és a DNS rekordok helyesen mutatnak a szerverünkre, a Dokploy automatikusan kér majd egy HTTPS tanúsítványt a Let's Encrypt szolgáltatótól (erről bővebben a következő pontban). Fejlesztési vagy tesztkörnyezetben előfordulhat, hogy ideiglenesen nem szükséges HTTPS (pl. ha csak gyorsan meg akarunk nézni valamit HTTP-n keresztül), ilyen esetben ezt OFF-ra hagyhatjuk. **Fontos:** A Dokploy által biztosított ingyenes *traefik.me* domainek alapból **nem támogatják az automatikus HTTPS-t** – ezek csak HTTP-n működnek, kivéve ha kézzel feltöltünk és beállítunk hozzájuk egy tanúsítványt ³. Tehát ha ingyenes domainnel dolgozunk, hagyjuk a HTTPS kapcsolót *kikapcsolva*, vagy külön konfiguráljunk tanúsítványt hozzá.

- **Certificate Provider (Tanúsítvány):** Itt választhatjuk ki, honnan származzon a HTTPS tanúsítvány. Alapértelmezésben a **Let's Encrypt** opciót szokás választani, ami azt jelenti, hogy a Traefik **automatikusan megpróbál egy hitelesítésszolgáltatótól (Let's Encrypt) érvényes TLS tanúsítványt szerezni** a megadott domain névre ¹⁶. A Let's Encrypt egy ingyenes, automatizált tanúsítványkiadó; a Dokploy integrációja révén az egész folyamat automatikus – a domain-hitelesítés (HTTP-01 challenge) és a tanúsítvány letöltése, telepítése megtörténik magától, feltéve hogy a domain név már a szerverünkre mutat. (Ennek feltétele, hogy a 80-as porton elérhető legyen a Traefik a hitelesítés idejére, mivel a Let's Encrypt a domainhez tartozó `.well-known` URL ellenőrzésével igazolja a tulajdonjogot.) Alternatívaként választható a *None* (Nincs tanúsítvány) opció is, ekkor a Dokploy nem kér tanúsítványt – ezt tipikusan akkor használjuk, ha

nem akarunk vagy nem tudunk HTTPS-t beállítani. Például Cloudflare **Flexible SSL** mód esetén (amikor a Cloudflare csak a saját szerverei és a böngésző között használ HTTPS-t, a mi szerverünkkel már nem), a Dokploy-ban **ki kell kapcsolni a HTTPS-t és a Certificate mezőt "None"-ra állítani**, hogy a Cloudflare HTTP-n érhesse el az origin szervert konfliktus nélkül ¹⁷. (Megjegyzés: a Cloudflare *Full (Strict)* módhoz viszont engedélyeznünk kell a HTTPS-t és a Let's Encrypt tanúsítványt, hiszen ott a Cloudflare is elvárja a hitelesített kapcsolatot a mi szerverünk felé is ¹⁸ ¹⁹.)

HTTPS és az automatikus Let's Encrypt tanúsítvány

Ha a fenti úrlapon a **HTTPS kapcsolót bekapcsoljuk és a Certificate mezőben a "Let's Encrypt" lehetőséget választjuk**, a Dokploy a Traefik segítségével automatikusan gondoskodik a tanúsítvány igényléséről és kezeléséről. **Hogyan működik ez?** Miután létrehoztuk a domaint az alkalmazáshoz, a Traefik megpróbálja elvégezni a Let's Encrypt-tel a domain érvényesítését. Ez általában egy **HTTP-01 challenge** útján történik: a Let's Encrypt egy speciális token-fájlt kér le a `http://<a-domainunk>/.well-known/acme-challenge/` útvonalon keresztül. Ezt a kérést a Traefik fogadja, és mivel tudja, hogy tanúsítványt kell szereznie, a megfelelő válasszal szolgál a Let's Encrypt felé (a háttérben a Dokploy/Traefik kezeli ennek technikai részleteit). Fontos, hogy a domain nevünk **már a megfelelő szerverre mutasson** (pl. A rekordunk rendben legyen), különben a hitelesítés sikertelen lesz. Sikeres challenge esetén a Let's Encrypt kiad egy **Domain Validated (DV) TLS tanúsítványt**, amit a Traefik eltárol és onnantól kezdve az adott domainhez használ. A Traefik gondoskodik a tanúsítvány **automatikus megújításáról** is (a Let's Encrypt tanúsítványok 90 napig érvényesek, de a megújítás általában a lejárat előtt ~30 nappal megtörténik automatikusan).

A felhasználó szempontjából mindez **teljesen automatikus**: ha helyesen állítottuk be a domain nevet és bekapcsoltuk a HTTPS + Let's Encrypt opciót, akkor a domainünk rövid időn belül HTTPS alatt lesz elérhető, érvényes tanúsítvánnyal. A Dokploy UI-ban valószínűleg látni fogunk visszajelzést, ha a tanúsítvány igénylése sikeres volt (vagy hibát, ha nem). Még egyszer kiemelendő, hogy ingyenes *traefik.me* domain használatakor alapból nem történik automatikus tanúsítvány igénylés (hiszen arra **None**-t állítunk be), azonban **bármikor átállhatunk saját domainre**, ahol már a fenti mechanizmus működik.

Domain név DNS konfigurációja (A rekord vs. CNAME Traefik esetén)

Miután áttekintettük az űrlap mezőit, nézzük meg, hogyan kell **DNS oldalról beállítani a domain nevünket**, hogy a Traefik reverse proxy valóban meg is kapja a forgalmat.

Alapvető DNS-beállítás – A rekord: A leggyakoribb eset, hogy van egy saját domain nevünk (pl. `sajatdomain.hu`), és ezen szeretnénk futtatni az alkalmazást (vagy egy aldomainjén). Ilyenkor a domain szolgáltatónk DNS kezelő felületén létre kell hoznunk egy **A rekordot**, amely az adott domain nevet (vagy aldomaint) a Dokploy/Traefik szerverünk IP-címére irányítja ²⁰. Például: ha az alkalmazást az `api.sajatdomain.hu` címen akarjuk elérni, akkor az **A rekord** típusa legyen **A**, a neve `api` (így lesz belőle `api.sajatdomain.hu`), értéke pedig a szerverünk nyilvános IPv4 címe (pl. `1.2.3.4`) ²⁰. Mentés után tipikusan néhány percen (vagy max órákon) belül a DNS frissül, és az `api.sajatdomain.hu` már a mi szerverünkre fog mutatni. Ezután a Dokploy UI-ban a fent részletezett módon hozzuk létre a domaint (Host mezőbe `api.sajatdomain.hu`, Path `/`, stb.), kapcsoljuk be a HTTPS-t, Let's Encryptet, és hozzárendeljük az alkalmazáshoz ²¹. Ha mindent jól csináltunk, pár percen belül az alkalmazásunk élő lesz ezen a címen, és működik is.

WWW aldomain (CNAME) beállítása: Gyakori igény, hogy egy weboldal mind a `www.sajatdomain.hu`, mind a `sajatdomain.hu` alatt elérhető legyen. Ennek egyik módja, hogy a két nevet külön domainként felvesszük a Dokploy-ba és megfelelően irányítjuk őket. Egyszerűbb azonban az, ha az egyik a másikra át van irányítva. Például megtehetjük, hogy a `www` aldomain egy **CNAME rekorddal** a root domainre mutat. A DNS-nél ilyenkor: **Típus:** CNAME, **Név:** `www`, **Érték:** `sajatdomain.hu` ²². Ezzel a `www.sajatdomain.hu` valójában a `sajatdomain.hu`-ra fog feloldani. A Dokploy-ban ezután létrehozhatunk egy domain bejegyzést `www.sajatdomain.hu` Hosttal is (ugyanarra az alkalmazásra), vagy használhatjuk a Dokploy **Redirects** funkcióját: pl. az alkalmazás -> Advanced -> Redirects résznél beállítható egy előre definiált szabály "`www to non-www`" ²³, így a Traefik automatikusan átirányítja a `www.` kezdetű kéréseket a `www` nélküli domainre (vagy fordítva, igény szerint). Összefoglalva: **A rekordot** használunk minden olyan domainhez/ aldomainhez, amely közvetlenül egy IP-címre mutat, **CNAME rekordot** pedig akkor, ha egy adott nevet egy másik hostnévhez akarunk kötni. Mivel a mi esetünkben a Traefik saját szerveren fut, általában a root domain(ek)et A rekorddal az IP-re irányítjuk, és csak kiegészítő aliasokra (pl. `www`) használunk CNAME-et. (Megjegyzés: a *traefik.me* domain használatakor nincs szükség semmilyen DNS beállításra a részünkről – a generált hostnév kódolja az IP-címet, és a traefik.me szolgáltatás automatikusan arra oldja fel ².)

Dokploy használata Docker Swarm és standalone Docker környezetben

A Dokploy rugalmasan működik **egy szerveres (standalone Docker)** és **több szerveres (Docker Swarm klaszter)** környezetben is. Fontos tudni, hogy **akármelyik módban használjuk, a domain beállítások és a Traefik proxy szerepe alapvetően ugyanaz marad**, azonban a háttérben a működés kissé eltérhet.

- **Standalone (egyszerű Docker) eset:** Ilyenkor a Dokploy egyetlen szerveren fut, és ezen a gépen fut a Traefik is, illetve az összes konténerizált alkalmazás. A fenti domain konfigurációk mind erre az egy gépre mutatnak. Tehát az **A rekordjainknak a standalone szerver IP-címére kell mutatniuk**, mert a Traefik ott hallgat a 80-as és 443-as porton. Minden bejövő webes forgalom ezen a gépen keresztül megy, és a Traefik a beállított Host/Path alapján továbbítja a megfelelő konténer(ek) felé. A Dokploy UI-ban a domain hozzárendelésekor nem is kell foglalkoznunk azzal, melyik gépen fut az alkalmazás, hiszen csak egy van – automatikusan ezen érhető el.
- **Docker Swarm (több szerveres klaszter) eset:** A Dokploy lehetőséget ad arra, hogy **horizontálisan skálázzunk**, azaz több VPS-t/hostot vonjunk be egy klaszterbe ²⁴. Ilyenkor a Docker Swarm technológia alatt össze van kapcsolva a szerverek csoportja. **Hogyan kapcsolódik ehhez a domain kezelés?** A Dokploy telepítésekor a Traefik is integrálva van – Swarm módban általában a Traefik egy szolgáltatásként fut a klaszterben (például a manager node-on, vagy akár globálisan minden node-on). A lényeg, hogy a Traefik továbbra is centralizáltan kezeli a forgalmat. Tipikus beállítás, hogy a klaszter **belépési pontja** egy konkrét node (pl. a manager node) IP-címe, amire a domaineket irányítjuk. Tehát DNS oldalon itt is általában egy (vagy több) **A rekorddal** a klaszter bejárati pontjára mutatunk. Ha például a manager node IP-je `5.6.7.8`, akkor az A rekord értéke `5.6.7.8`. Amikor egy kérést kap a Traefik (a manager node-on), a Docker Swarm hálózatán belül **tovább tudja irányítani** a megfelelő konténerhez akkor is, ha az esetleg egy másik gépen fut ²⁵. A Traefik integrálódik a Docker Swarm service discovery rendszerével, így ismeri a konténereket/szolgáltatásokat a klaszter minden node-ján ²⁶. A Dokploy UI-ban a domain hozzárendelése és az alkalmazások deployolása hasonlóan történik, mint egy gépnél – azzal a különbséggel, hogy előfordulhat, meg kell mondanunk, melyik szerveren (vagy Swarm service-ként) fusson az adott alkalmazás. A **Traefik + Swarm** páros

gondoskodik arról, hogy a forgalom eljusson a megfelelő konténerhez a klaszteren belül. Ha nagy rendelkezésre állást szeretnénk, lehetőség van Traefik-et több node-on is futtatni és egy külső terheléelosztót (vagy DNS round-robin A rekordokat több IP-vel) használni, de alaphelyzetben a Dokploy Swarm cluster esetén egy entypoint elegendő, mivel Traefik maga is tud terhelést osztani a konténerek között. **Összefoglalva:** a Docker Swarm környezetben a domain név beállításai ugyanúgy néznek ki (A rekord a megfelelő IP-re), a Dokploy/Traefik pedig a háttérben intézi, hogy a kérés a megfelelő node-on lévő konténerig jusson el. Felhasználóként nincs további extra teendők, csak gondoskodjunk arról, hogy a **Traefik fut és elérhető a külvilág felől** minden olyan node-on vagy IP-címen, amit a DNS rekordokban megadtunk.

Végül, függetlenül attól, hogy standalone vagy Swarm módon futtatjuk a Dokploy-t, **mindig ellenőrizzük a beállításokat:** a DNS rekordok megfelelően mutatnak-e a kiszolgálóra, a konténer port helyesen van-e megadva, fut-e az alkalmazás konténere, és szükség esetén nézzük meg a Traefik illetve alkalmazás logjait hibakereséskor ²⁷. Ha mindent helyesen konfiguráltunk, a Dokploy segítségével néhány kattintással megbízhatóan futtathatunk alkalmazásokat saját domain név alatt, akár egyetlen szerveren, akár egy többgépes Docker Swarm klaszterben.

Források:

- Dokploy dokumentáció – Domain beállítása (mezők leírása) ²⁸ ²⁹
- Dokploy dokumentáció – Internal Path és Strip Path magyarázata ⁵ ⁸
- Dokploy dokumentáció – Konténer port és megjegyzések a használatáról ¹⁵
- Dokploy dokumentáció – Ingyenes traefik.me domain használata ³⁰ ³
- Dokploy dokumentáció – Saját domain (DNS A rekord) beállítása, HTTPS és Let's Encrypt használata ³¹ ³²
- Traefik.me információk – automatikus wildcard DNS működés ²
- Dokploy dokumentáció – Cloudflare használata esetén szükséges beállítások ¹⁷ ¹⁹
- Dokploy dokumentáció – Docker Swarm klaszter és Traefik integráció ²⁵ ²⁶

¹ ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ ¹⁰ ¹¹ ¹⁴ ¹⁵ ²² ²³ ²⁷ ²⁸ ²⁹ Domains | Dokploy

<https://docs.dokploy.com/docs/core/domains>

² traefik.me

<https://traefik.me/>

¹² ¹³ ³⁰ Generated | Dokploy

<https://docs.dokploy.com/docs/core/domains/generated>

¹⁶ ²⁰ ²¹ ³¹ ³² Others | Dokploy

<https://docs.dokploy.com/docs/core/domains/others>

¹⁷ ¹⁸ ¹⁹ Cloudflare | Dokploy

<https://docs.dokploy.com/docs/core/domains/cloudflare>

²⁴ ²⁵ Cluster | Dokploy

<https://docs.dokploy.com/docs/core/cluster>

²⁶ Architecture of Dokploy | Dokploy

<https://docs.dokploy.com/docs/core/architecture>