Enterprise
Xpandrive
virtuális storage rendszer

Enterprise
Xpandrive virtuális storage rendszer

FELHASZNÁLÓI
KÉZIKÖNYV

Bevezető

Az ExpanDrive egy adattároló megoldás, amely szolgáltatásban gazdagabb és költséghatékonyabb alternatívát kínál a nagyvállalati adattároló megoldások tekintetében a jelenleg ismert termékeknél.

Az ExpanDrive a jelenleg ismert vezető gyártók kész, hardveres megoldásainak szolgáltatásainak egy részét létező nylít forrású megoldásokkal helyettesíti, illetve ezeket kis részben saját fejlesztésű, szintén nyílt forrású komponensekkelegészíti ki. A megvalósítani kívánt szoftverrendszer nagyon sokféle hardveren - akár heterogén rendszerben is - használható, ezzel megszünteti a hardver vendor lockin problémát.

Az ExpanDrive egyik nagyon fontos tulajdonsága, hogy nagyrészt jól ismert, nyílt forrású megoldások alkalmazásával épül fel. Az adattovábbítás és tárolás minden lépése nyílt forrású szoftvereken megy keresztül. Ezzel a célunk az, hogy átlátszó, megbízható, és biztonságos rendszert építsünk fel.

Az ExpanDrive részben egy fejlesztési keretrendszer is. Ez a keretrendszer lehetővé teszi speciális célokat szolgáló adattároló megoldások implementálást. Lehetséges éldául olyan tárrendszerek megvalósítása, amelyek különösen jól teljesítenek backup megoldásként, CDN-ek háttértárjaként, vagy elosztott fájlrendszerek alatt.

AZ EXPANDRIVE FŐBB TULAJDONSÁGAI:

- Bármilyen méretű és darabszámú blokkos eszköz mutatása az operciós rendszer felé - csak a használt blokkok foglalódnak le a tárolórendszeren.
- Háttértáranként változtatható blokkméret.
- Verziókezelt blokk tárolás. A háttértár bármelyik időbeni állapota bármikor rendelkezésre áll - “continuous snapshotting”.
- Shared storage funkcionalitás (pl.: cluster fájlrendszerekhez, Oracle RAC megoldáshoz)
- Shadow copy szolgáltatás - offsite backups, remote failover site replikáció
- Load balancing - új storage node hozzáadásával sebességben és kapacitásban skálázódik a rendszer
- Magas rendelkezésre állás - n fizikai diszk vagy node kiesését tűrő megoldás.
- Redundancia - blokkos eszközönként megadható, hogy hány példányban és hol (multi site működés esetén) legyenek tárolva az adatok
- Kernel driver linuxhoz
- Adminisztrációs command line parancsok
- Adminisztrációs web user interface

Architektúra

A jelen dokumentációval leírt rendszer két nagy részre bontható. Mindkettő több számítógépből álló klaszter is lehet. A rendszer úgy építjük fel, hogy az eredmény minél több számítógépet magában foglaló, minél inkább elsztott, skálázható és jó hibatűrésű lehessen.
A megoldás egyik része egy adattároló felhő, tárklaszter, ami egy MongoDB, vagy Riak klaszter. Ez a klaszter szükség esetén gyorsan bővíthető újabb csomópont (esetünkben egy számítógép) hozzáadásával. Az ExpandDrive megoldás egyik lényege, hogy ilyen tároló klaszterek viszonylag gyorsan illeszthetőek legyenek minél kevesebb fejlesztéssel. Ennek következtében a tárklaszter szerepét bármilyen megoldás be tudja tölteni, ami képes 512 - 4096 byte-os blokkok tárolására, és azok visszakeresésére egy kulcs alapján, ezek törlésére és módosítására.

A tárklaszter igen szorosan csatoltnak mondható, mivel a komponensek nagyon hasonlóak, és egyes párok, vagy csoportok szoros függőségben állhatnak egymástól. Normális esetben konfigurációjuk keveset változik, és külső beavatkozásra is kevés lehetőség van. (Itt csupán a normális működésről van szó, a rendszer nyílt forrású, így természetesen tetszőlegesen módosítható.)
A másik rész a kliensek felhője, ami használja a tárklaszter szolgáltatásait. A kliens klaszter egy lazán csatolt formáció, mivel még a kliens csomópontok célja, és együttműködésük módja sem definiált, az egyetlen közös dolog bennük az, hogy ugyanazon rendszerben kezeljük őket, és használhatják a tárklaszter szolgáltatásait.

A fent említett elemek a rendszer működése szempontjából kritikusnak tekinthetőek, mivel ezen rendszerek meghibásodása a szolgáltatás kiesését, katasztrófát jelent. Ezeket a részeket ezért úgy kell felépíteni, hogy egy elemük (egy hardverelem, mint egy merevlemez, vagy akár egy egész csomópont) kiesése ne okozza a klaszter szolgáltatásainak kiesését, vagy a szolgáltatások minőségének jelentős romlását, illetve a klaszter bővítése, vagy a kiesett elemek pótlása miatt a klaszer által elvégzendő plusz műveletek ne terheljék meg azt annyira, ami a működésben problémát okozhatna.

Ezen két fő komponens problémamentes együttműködését, és a karbantartási munkák segítését és automatizálását egyéb komponensek végzik. Ezen komponensek meghibásodása nem okoz szolgáltatáskiesést; sem a kliens csomópontok, sem a tárklaszter működése nem torpanhat meg. Így például a háttértárakat kezelő webes kezelőfelület elérhetetlensége nem szabad, hogy befolyásolja a klienseket a háttértárak elérésében.

A tár- és kliensklasztert segítő komponensek a következőek:

A rendszer elemeinek monitorozását egy Zabbix kiszolgáló végzi. Ez a komponens a megfigyelt csomópontokra telepített ügynökök segítségével gyűjt adatokat, tárolja, megjeleníti, és képes riasztások küldésére, vagy egyébb automatizált feladatok elvégzésére.

Az egyes csomópontok konfigurációjának kezelésére **Chef**-et alkalmazunk. A **web-es adminisztrációs felület**, illetve az általa nyújtott **API** a chef szerveren végzi a módosításokat, amiket a csomópontok alkalmaznak.

A webes adminisztrációs felület (web admin) végzi az új csomópontok felvételét a tár- vagy kliens klaszterekbe is. Mindkét esetben csupán egy SSH hozzáférést kell biztosítani egy RSA kulcs segítségével, a többi teendőt a web admin, illetve később a chef automatikusan elvégzi.

Kernel komponensek

A kernel driver felelős a blokkos eszköz (storage) létrehozásában és elérhetővé tételében az OS és az alkalmazások számára. A kernel driver szabványos blokkos eszközt generál, a TRIM hívásokat támogatja. A beérkező IO hívásokat az userspace daemon felé közvetíti egy speciális karakteres eszközön keresztül.

Felhasználói módú komponensek

Háttértár megoldás
A háttértár megoldás (storage backend) többféle adatbázis (jellemzően kulcs-érték tár) lehet.

A háttértár megoldás feladata, hogy a blokkos eszközök adatait tárolja. Használatánál ügyelni kell arra, hogy a nagy sávszélesség, és kis késleltetési idő megvalósítható legyen, illetve arra, hogy a rendszer skálázható legyen mind méretet, mind a kiszolgálható kliensek / szálak számát tekintve.

Felhasználói módú háttérfolyamat
Feladata hogy a beérkező IO kéréseket a megfelelő storage backend felé irányítsa, azaz tartsa a kapcsolatot a kernel driver és a storage backend között. A daemon különböző pluginekkel más-más célnak (felhasználási karakterisztikának) megfelelő storage backend felé képes irányítani az IO hívásokat.

Szoftverkörnyezet

A rendszer első verziója linux csomagokban érkezik, melyekkel a kernel driver, az userspace daemon, az adminiztrációs parancsok, a webes adminisztrációs felület, a monitoring szolgáltatások illetve a dokumentációk kerülnek fel a célgépekre.
Az adminisztációs parancsok illetve a felület szolgáltatásai megegyeznek a storage kezelését illetően. Ezen felületeken a storage világban megszokott szolgáltatások elérhetőek, mint például a storage alaműveletek (létrehozás, módosítás, eltávolítás), a mentések kezelése illetve a storage pool állapotának kijelzése.

Az adminisztrációs csomag része a rendszer állapotát ellenőrző szoftverkörnyezet, melynek szolgáltatásai:

- segítségével a rendszer üzemeltetői adatokat kapnak a rendszer működéséről
- riasztások küldése az üzemeltetők számára

Management eszközök

A szoftverkörnyezetet az egyszerű kezelhetőség szemontjából különböző eszközökkel támogatjuk, melyek segítik a rendszer üzemeltetését, frissítését, illetve testreszabását az adott felhasználási terület szempontjából.

A rendszerüzemeltetők támogatását a Chef, Zabbix illetve egyedi parancssoros és webes adminisztrációs eszközökkel biztosítjuk.
Az egyes modulok integrálását illetve fejlesztésével kapcsolatos irányelveket a Management eszközök pont alatt tárgyaljuk.
A következő fejezetek kifejtik a most vázolt komponenseket.

Kernel komponensek

Az ExpanDrive szoftverrendszer legérzékenyebb része a kernelben futó kód. A Linux kernel monolitikus, ebből adódóan az egyes részei viszonylag erős csatolásban vannak egymással, és minden része magas privilégiumszinten fut, így szinte bármit megtehet a hardverrel és a futó folyamatokkal.

A SZOFTVERRENDSZER HÁROM FŐ RÉTEGBŐL ÁLL:

- Meghajtó - rendszermag szint
  - Általános célú
  - Nyílt forrású
  - Feladata az IO kérések delegálása
- Felhasználói szintű démon
  - IO kérések kiszolgálása
  - Kapcsolat a rendszermag és egy (vagy több) társzolgáltatással
- Társzolgáltatás
  - MongoDB

BDTUN DRIVER

A szoftverrendszer rendszermagkomponensét BdTun-nak nevezzük (Block Device TUNneling).
A bdtun egy kernel modul, ami Linux alatt a szokott módon, a szokott eszközökkel illeszthető az operációs rendszer magjába.

Felhasználói módú komponensek

Jelen szoftverrendszer úgy lett kialakítva, hogy a legtöbb komponense felhasználói módban (user space) fusson, ezzel csökkentve az előforduló hibák által okozott leállások valószínűségét.

Az alábbiakban részletezzük, hogy az egyes felhasználói szintű komponensek milyen szerepeket játszanak, és hogy milyen működésbeli tulajdonságok garantálják a stabilitást, újraindthatóságot és hibatűrést.

TÁR ILLESZTŐ FOLYAMAT (STORAGE DAEMON)

Tulajdonképpen ez az a komponens, ami hagyományos esetben nagyrészt a rendszermagban foglalna helyet. Ez a rendszer egyik kulcs-fontosságú része: részt vesz hálózati kommunikációban, ami természeténél fogva megbízhatatlan. Nagy mennyiségű adatot mozgat meg, bizonyos alkalmazások esetén akár konverzót, vagy tömörítést is végez.
A storage daemon feladata kapcsolatot tartani a rendszermag és tárklaszter (vagy bármilyen blob tár megoldás) között.

Parancssoros felület

A parancssoros felület a rendszermagban futó komponens vezérlésére való. Ezzel az eszközzel lekérdezhetjük az eszköz által kezeltt blokkos eszközöket, létrehozhatunk újat, vagy törölhetünk, átméretezhetünk meglévőeket.

MongoDB storage backend

A MongoDB storage backend egy mongodb klaszter. A MongoDB jó választás BLOB tárolásra, mivel hálózati protokollja bináris, és semmilyen konverziót nem követel meg. Megfelelő hardver- és szoftverkörnyezetben akár az is megoldható, hogy egy bináris adatot tartalmazó puffer tartalma a felhasználói folyamattól indulva, a kernelen áthaladva, majd végül az ExpanDrive felhasználói módú rendszerfolyamathoz kerülve úgy jusson a hálózatra, hogy közben egyszer sem történik másolás, vagy konverzió, így a CPU igény a pufferméttel O(1)-ben skálázódik, illetve gyakorlatilag is kevés.

A MongoDB képes bonyolult módon indexelni, ez a képessége bőven elég arra, hogy verziókezelést lehessen vele megvalósítani. Éppen ezért a MongoDB backendet elsősorban verziókövetett kötetek megvalósítására ajánljuk.

Szoftverkörnyezet

Management eszközök

CHEF

A Chef az Opscode által fejlesztett és támogatott rendszer management környezet, amelyet az ExpanDrive arhitektúrájába integráltuk testreszabva azt, hogy a tervezett funkcionalitást ellássa.

Chef szerver

A chef egy kliens-szerver alapú megoldás. A szerver oldali fogalmak az alábbiak.

Search

A search szolgáltatással kereshető a konfigurációs management adatbázisa. Ezzel a szolgáltatással alkalmazhatunk szűrőket, hogy adott folyamatot mely nodeokon szeretnénk futtatni.

Manager

A manager a webes kezelőfelület. A webes kezelőfelületet ingegráljuk az ExpanDrive webes admin interface-e alá. Ezt a manager API hívásain keresztül tesszük.

Node objektum: attribútumok

Az attribútumok azon nodeokhoz köthetők, amelyeket a rendszer kezel, esetünkben az ExpanDrive storage nodejai, fizikai gépei.
Az attribútumokat a nodeon futó chef kliens definiálja, amely felülírható receptekből, roleokból illetve környezetek (environments) megadásával. Az attribútum listát a kliens oldali (chef-client) futás kezdetén a kliens állítja össze. Amennyiben az előző futáshoz képest változások történtek a kliens oldal végzi el az esetleges módosításokat az adott node(ok)on.

Node objektum: run-list

A run-list egy szerveren adott node mellé tárolt feladatlista. Ez a lista adja meg a pontos sorrendjét az egyes nodeon (vagy node csoporton) futtatandó feladatoknak, módosítandó konfigurációknak. A knife kliens paranccsal módosítható a szerverhez csatlakozva.

Policy

A policy konfigurációs objektum segítségével képezzük le az üzleti illetve üzemeltetési elvárásokat illetve workflow-t ami alapján tároljuk az egyes nodeok vagy node csoportok konfigurációját.

Policy objektum: Role

A role objektumban tárolható az egyes funkció ellátásához szükséges recept illetve alapértelmezett attribútum környezet. A role objektum tartalmazhat további role objektumokat is.

A role objektum létrehozása: knife create role <rolename>

A ROLE OBJEKTUM SZERKEZETE AZ ALÁBBI:

```
{
  „description": „",
  „json_class": „Chef::Role",
  „run_list": [

  ],
  „default_attributes": {
  },
  „override_attributes": {
  },
  „name": „new",
  „env_run_lists": {
  },
  „chef_type": „role"
}
```

Policy objektum: environment

Az environment objektumot használjuk arra, hogy a nodeokat a fejlesztési workflow szerint megkülönböztessük egymástól. Létrehozunk development, staging, testing illetve production környezeteket, amelyekkel szabályozzuk, hogy környezettől függően milyen verziójú recepteket használunk. Ezzel teszteljük az új verziókat a fejlesztés során.

Environment létrehozása: knife environment create <envname>

AZ ENVIRONMENT STRUKTÚRÁJA:

```
{
  „json_class": „Chef::Environment",
  „chef_type": „environment",
  „description": „",
  „name": „new",
  „override_attributes": {
  },
  „default_attributes": {
  },
  „cookbook_versions": {
  }
}
```

A cookbook objektum létrehozása:

A cookbook az alapvető konfigurációs objektum. Ez tartalmazza a konkrét leírását azon komponenseknek amelyeket egy adott expandrive nodeon managelünk. A pontos listát a Chef receptek témakör tárgyalja.

A cookbook az alábbi elemeket tartalmazza: knife cookbook create <cookbook name>

A COOKBOOK AZ ALÁBBI ELEMEKET TARTALMAZZA:

- cookbook attribútumok (ez felülírhatja a node szintű attribútúmokat)
- definíciók, amellyekkel létrehozhatunk újra felhasználható
- fileok
- libraryk
- receptek
- templatek
- verziók
- metadata

Chef receptek

A receptekkel definiáljuk azon rendszerparamétereket amelyeket a management rendszer felügyel. Ebbe beletartozik minden ami a "bare-metal"-"minimal-os" telepítés után telepítendő és/vagy konfigurálandó a storage nodeokon.

OS szintű konfigurációk

Az OS szintű konfigurációk az alábbi feladatokat látják el:

1. Minden nodeon

- **Network konfigurációk**
  - dns szerverek
  - ip címek
  - gw
  - dns search path
  - nis konfiguráció
  - dhcp beállítások
- **Network time protocol (NTP)**
- **Chef client**
- **Monitoring agent (Zabbix)**
- **Frissítési csomagforrások (APT)**

2. Adminisztrációs nodeokon

- **Zabbix szerver**
- **MySQL szerver (zabbix konfigurációs backend)**
- **PostgreSQL szerver (expandrive storage konfiguráció store)**
- **Apache webszerver**
- **PHP**
- **ExpanDrive WebGUI**

Storage daemon konfiguráicó

A storage daemon konfigurációs recept tartalmazza a storage daemon telepítési definíciókat illetve a storage daemon konfigurációs fájlokat. A storage daemon konfigurációs fájl a /etc/bdtun.conf elérési útra kerül.

MongoDB storage backend konfiguráció

A mongoDB recept tartalmazza a mongodb telepítéséhez szükséges hivatalos APT repository-t (vagy a saját apt repositorynkat feltéve, hogy a mongodb forrásán is változtatunk), a mongodb start-stop scriptjeit illetve a mongodb külső konfigurációs fájlokat.
Ez utóbbiak a /etc/mongodb könyvtárban kerülnek tárolásra.

Chef kliens

A chef kliens komponens végzi el a tényleges node konfigurációs módosításokat a chef-server komponensben meghatározott definíciók alapján. A konfigurációs fájljai a /etc/chef alatt találhatóak.

Client.rb - a kliens konfigurációja

client.pem - az x509 certificate amelyel létrejön a kliens és a szerver közti biztonságos kapcsolat. Ez alapján történik a node azonosítása is a szerveren.

RENDSZERTELEPÍTŐ

A rendszertelepítő komponens végzi el a production rendszer cluster feltelepítését egy új környezetben.
A rendszertelepítő CD-ROM image illetve standard USB storage (pl. Pendrive) formátumban kell, hogy elérhető legyen.
A telepítés végigvisz az alapszintű rendszer beállításokon amelyek minimálisan szükségesek ahhoz, hogy a rendszer webGUI illetve cmdline interface-el konfigurálható legyen.

Boot image

A BOOT IMAGE FELÉPÍTÉSE:

1. boot manager
Boot manager szolgáltatásként a "syslinux" csomagot használjuk.
2. linux kernel
A telepítő rendszere linux kernelen fut.
3. initramfs

A kernel betöltése és elindulása után az initramfs mountolása következik. A mountolás után a /sbin/init fájl indításával indul el a telepítő. Az initramfs image tartalmazza az összes általunk supportált hardver drivereit.
Beállítások a telepítőben

A TELEPÍTŐ A KÖVETKEZŐ BEÁLLÍTÁSI LEHETŐSÉGEKET KÉRI BE ILLETVE ALKALMAZZA A TELEPÍTÉS SORÁN:

- node típusa (admin/storage)
- node neve
- hálózati beállítások
- ntp szerverek
- cluster neve

A cluster nevének bekérése után egy ellenőrzés következik, hogy a cluster létezik-e, vagy új cluster telepítéséről van-e szó. Előbbi esetben a node integrálódik a clusterbe az alábbi szolgáltatásokon való regisztrációval:

- Chef szerver
- Zabbix szerver
- ExpanDrive WebGUI

MONITORING ALRENDSZER

A rendszer üzemeltetés része, hogy van real-time monitoring illetve alerting alrendszer, hogy a rendszert üzemeltető csoport aktuális képet kapjon a rendszer tabilitási illetve teljesítmény adatairól, a mentés állapotáról, stb.

Az ExpanDrive szoftverkörnyezet monitoring alrendszeréül a "zabbix" monitoring megoldást választottuk. Ez egy kliens-szerver alapú opensource monitoring tool.

Zabbix szerver konfiguráció

A ZABBIX SZERVER A KONFIGURÁCIÓJÁT KÉT HELYEN TÁROLJA:

1. fájlrendszeren lévő konfigurációk

A fájlrendszeren az induláshoz szükséges alap konfigurációs fájlok találhatóak a /etc/zabbix könyvtár alatt.

A beállításhoz a zabbix_server.conf fájlt szükséges a környezetnek megfelelően beállítani.

A beállításhoz fontosabb paraméterek:

- backend adatbázis elérési adatai
- node neve
- teljesítmény paraméterek (szálak száma, timeoutok és egyéb limitek)

2. adatbázisban (MySQL) tárolt konfigurációk

Ezen konfigurációk tartalmazzák a monitorozandó nodeokat, az azokhoz tartozó itemeket illetve azok részletes definícióját.

Zabbix kliens konfiguráció

A zabbix kliens (agent) végzi az adatgyűjtést. Az összegyűjtött adatokat a zabbix szerver felé küldi el.

Az agent konfigurációja a /etc/zabbix/zabbix_agentd.conf fájlba kerül.

Mért rendszerparaméterek

A MONITORING RENDSZER AZ ALÁBBI MÉRENDŐ RENDSZERPARAMÉTEREKET MÉRI:

1. minden csomópoton

- kernel
- hálózat
- blokkos eszközök
- fájlrendszerek

2. admin csomópoton

- zabbix szerver állapota
- Apache webszerver állapota

3. storage csomópoton

- kernel driver állapota
- storage daemon állapota
- storage backend (mongoDB) állapota

Hibajelzések

Webes adminisztrációs felület

KOMPONENSEK

PHP

A PHP egy HTML-be ágyazható szkriptnyelv, szintaxisát a C, Perl és Java nyelvekből kölcsönözte. Eredetileg a szerzők egy nagyon egyszerű eszköznak szánták, ami a statikus HTML fájlokba egy kis dinamizmust csempészhet. A funkcionalitása ekkor hasonló volt az Apache HTTPD server side includes szolgáltatásához.

Az első verzió megjelenése óta sok idő telt el, és a PHP nagyon sokat fejlődött, közben igyekezve megtartani kompatibilitását a régi verziókkal. Ez a tény, és az, hogy a fejlesztők nem jeleskednek különösebben a programozási nyelvek és API-k írásának mesterségében azt eredményezte, hogy a PHP - bár teljesen használható - de sok szempontból mégsem a legjobb választás.

Valójában az egyetlen előnye, hogy nagyon elterjedt, ezért könnyű hozzá fejlesztési és üzemeltetési szakértelmet szerezni, gyakran igen kedvező áron. Az alacsony árnak azonban gyakran a szalértelem minősége látja kárát: mivel a PHP fejlesztés és üzemeltetés nagyon széles piac, ezért sok ár és minőségszint megtalálható, s természetesen a kettő erőteljesen.

A WEBALKALMAZÁS ÁLTAL KEZELT OBJEKTUMOK

Tár csomópontok

Egy tár csomópont egy - virtuális, vagy fizikai - számítógép, aminek szerepe az, hogy adatokat tároljon több számítógéppel együttműködve.

AZ ALÁBBI MŰVELETEK VÉGEZHETŐEK VELE:

- Hozzáadás tárklaszterhez
- Kivétel a tárklaszterből

Tárklaszterek

A tárklaszter számítógépek, "csomópontok" egy csoportja. A klaszter tagjai egymással együttműködve képesek adattárolásra oly módon, hogy egy, vagy - megvalósítástól függően - néhány csomópont kiesése nem okozhatja a klaszter egészének meghibásodását, így az adattárolási szolgáltatások nem állhatnak le, és ilyen kiesés nem okozhat adatvesztést.

A TÁRKLASZTEREKKEL A KÖVETKEZŐ MŰVELETEK VÉGEZHETŐK:

- Tárklaszter létrehozása
- Csomópont felvétele (MongoDB esetén csak replika set-be)
- Replika set felvétele (csak MongoDB esetén)
- Tárklaszter megszüntetése

Kliens csomópontok

A kliens csomópontok számítógépek, amik egy, vagy több tárklaszter szolgáltatásait igénybevehetik. A kliens csomópontok csoportokba rendeződnek. Egy kliens csomópont több csoport tagja is lehet.

AZ EGYES KLIENSEKEN A KÖVETKEZŐ DOLGOKAT VÉGEZHETJÜK EL:

- Kötet hozzáadása
- Kötet eltávolítása
- Csoportba felvétel
- Csoportból eltávolítás

Kliens csoportok

A kliens csoportok kliens csomópontok halmazai. Szerepük az, hogy megkönnyítsék a kliens csomópontokhoz való felhasználói hozzáférés szabályozását. Egy kliens csomópont több csoport tagja is lehet.

A CSOPORTOKON VÉGEZHETŐ MŰVELTEK:

- Létrehozás
- Törlés
- Kliens felvétele
- Kliens törlése
- Csoport átnevezése
- Felhasználó felvétele
- Felhasználó törlése

Felhasználók

Ha felhasználókról írunk, az ebben a fejezetben mindig az adminisztrációs webalkalmazás felhasználóit jelenti.
A felhasztnálók beléphetnek, bizonyos adataikat módosíthatják. Hozzáférhetnek klienscsoportokhoz, illetve az egyes kliensekhez, tárklaszterekhez, vagy más felhasználókhoz, és azokon műveleteket végezhetnek el.

Minden felhasználóhoz nyilván van tartva, hogy hozzáférhet-e egy adott tárklaszterhez, illetve klienscsoporthoz. Egy csoporthoz, vagy klaszterhez való hozzáférés egyben a tartalmazott csomópontokhoz való hozzáférést is jelenti. A felhasználó egy speciális tulajdonsága, hogy "supervisor"-e. Egy supervisor minden klienscsoporthoz és tárklaszterhez hozzáfér, és képes felhasználókat létrehozni, és bárki adatait módosítani. A supervisor jogokkal nem rendelkező felhasználók csak a számukra külön kijelölt csoportokat és klasztereket érik el, valamint nem tudnak új felhasználót létrehozni, vagy magunkon kívül más felhasználó adatait módosítani.

A FELHASZNÁLÓKON AZ ALÁBBI MŰVELETEK VÉGEZHETŐEK:

- Belépés
- Kilépés
- Létrehozás
- Törlés
- Zárolás
- Jelszó módosítás
- Tulajdonságok módosítása
- Supervisor
- Név
- E-mail cím
- Kliens csoport hozzárendelése / eltávolítása
- Tárklaszter hozzárendelése / eltávolítása

Kötet

A kötet egy, a kliensek által használható - esetünkben virtuális - tárolóeszköz. A kliens számítógép (vagy számítógépek) létrehozhat rajta partíciókat, fájlrendszereket, illetve tetszőlegesen felhasználhatja.

Egy kliens csomópont több kötetet is használhat, és egy kötet több kliens csomóponthoz is tartozhat. Ennek hasznosságát beláthatjuk ha az elosztott fájlrendszerekre, így például az OCFS2-re gondolunk, ahol szügség van arra, hogy az azt használó kliensek számára rendelkezésre álljon egy blokkos eszköz, amit egy mechanizmus egymással szinkronban tart (mondjuk úgy, hogy azok ugyanazon tárterületre mutató iSCSI eszközök).

A KÖTETEKKEL AZ ADMINISZTRÁCIÓS FELÜLETEN A KÖVETKEZŐK VÉGEZHETŐEK EL:

- Kötet létrehozása
- Kötet eltávolítása
- Kötet formázása: fájlrendszer létrehozása partíciós tábla nélkül
- Kötet összenyomása, vagy lapítása (squash): régi verziók törlése az adatok megtartásával
- Kötet kliensekhez rendelése
- Kötet átméretezése

BEJELENTKEZÉS

A Webes kezelő felület minden alkalommal, amikor nem vagyunk bejelentkezve, vagy az adott munkamenet lejárt, a bejelenetkező ablakot nyitja meg. Itt a rendszerben regisztrált E-mail címünkkel és jelszavunkkal tudunk belépni a rendszerbe.

Egy e-mail címmel csak egy felhasználói fiók regisztrálható. Az e-mail cím egyedi azonosító, ami különösen azért fontos, mert az elfelejtett jelszó esetén történő jelszócserekor erre az e-mail címre megy ki az egyszeri azonosításra szolgáló e-mail, benne az azonosításra szolgáló linkkel.

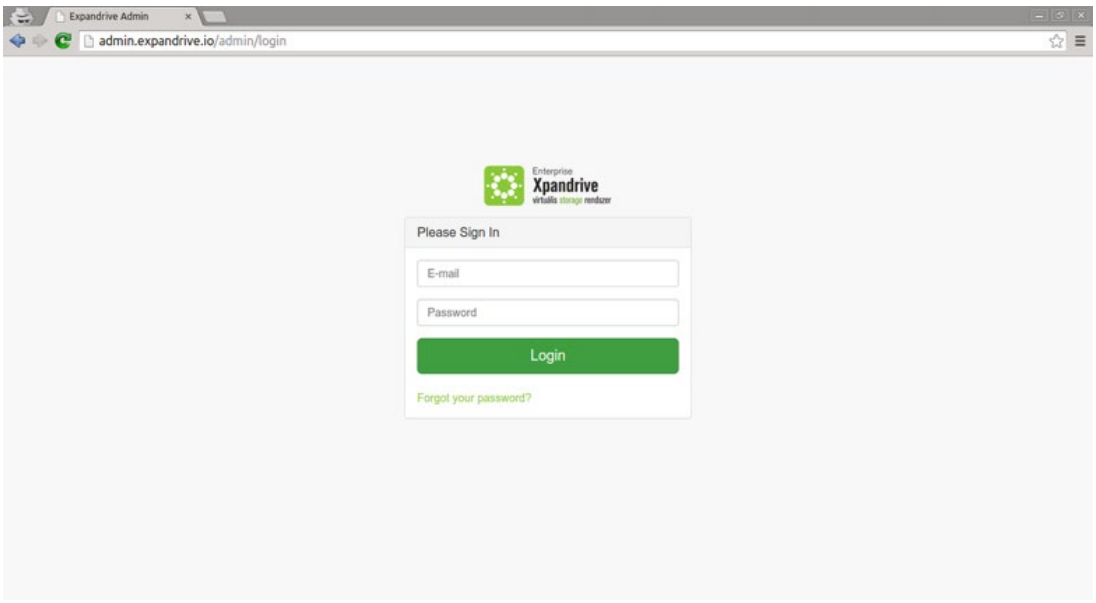
A jelszót A PKCS #5 v2.0 szabványban definiált PBKDF2 függvénnyel kell kezeljük, kellően megnehezítve ezzel a jelszavak megismerését akkor is, ha egy támadó a jelszavak tárolt változatait megismerte, illetve megnehezítve a jelszavak kitalálását még akkor is, ha a támadó egyéb erőforrás-korlátozó óvintézkedéseket megkerülve a szerver teljes számítási kapacitását a jelszavak kitalálására tudja fordítani. Az adatok kitöltése utáni OK gomb, vagy a billentyűzet Enter gombjának megnyomása után a felület elküldi az adatokat a web szervernek.

A SZERVER ELLENŐRZI AZ ADATOKAT, ÉS AZ EREDMÉNYTŐL FÜGGŐEN TÖBB DOLOG IS TÖRTÉNHET:

1. A rendszer nem talált olyan sort az adatbázisban,ami az elküldött E-mail címmel azonosítható lenne, ebben az esetben a kontroller nem is halad tovább és újra meghívja a bejelentkező felületet. Az így betöltött bejelentkezési felületen a felhasználó láthatja, hogy az általa megadott E-mail cím nem megfelelő.

2. A rendszer megtalálja a megfelelő sort az adatbázisban, és megpróbálja összehasonlítani a tárolt, és a küldött jelszót. Ha az összehasonlítás nem sikeres, hibával tér vissza. Az így betöltött ban az esetben a kontroller osztály újra a bejelentkezési felület meghívásávbejelentkezési felületen a felhasználó láthatja, hogy az általa megadott jelszó nem megfelelő.

Sikeres azonosítás után a rendszer tárolja az adatbázisban az aktuális belépés időpontját, az IP címet, a session id, az user-agent header tartalmát, majd a kontroller osztály meghívja a rendszer összegző felületét, mely bejelentkezett állapotban a főoldalt is jelenti egyben. A bejelentkező felületen biztosítani kell egy linket, ami az elfelejtett jelszó esetén tud segítséget nyújtani a felhasználónak.



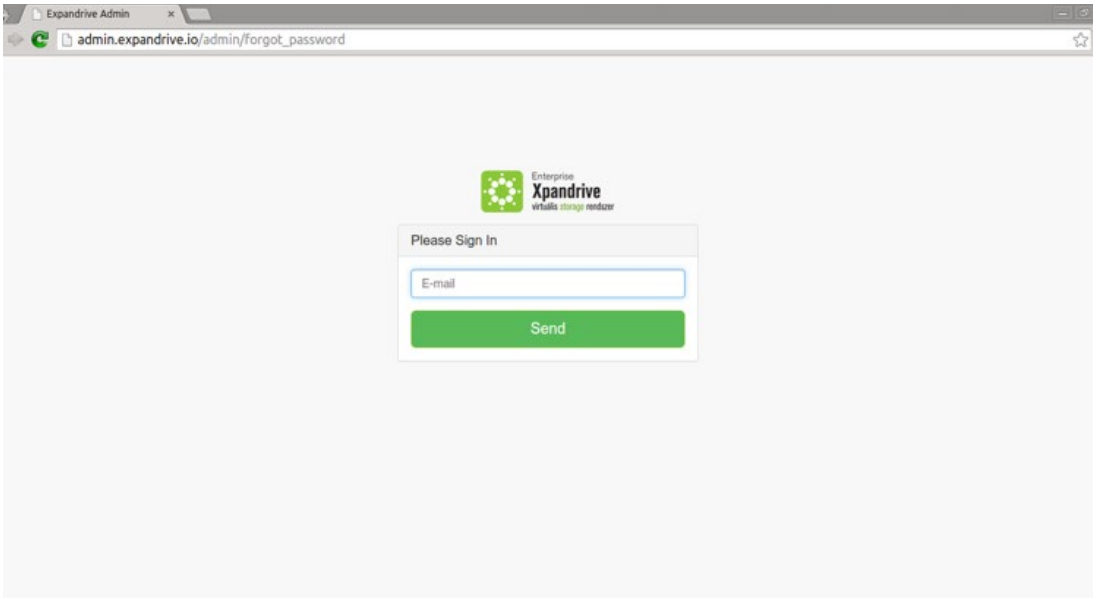
ELFELEJTETT JELSZÓ

Az elfelejtett jelszó felület az egyik, amely arra szolgál, hogy új jelszót kérjünk magunknak. Ezen a felületen az E-mail címünket megadva tudjuk kezdeményezni az új jelszó beállításának folyamatát. Az E-mail címet beírva, majd az "OK" gombra nyomva egy POST kérésben elküldjük a szervernek a beírt címet, ezt követően a kontroller előkeresi az adatbázisban az E-mail címhez tartozó felhasználót. Amennyiben megtalálja a rendszerben az adott sort, úgy generál egy speciális, 1 napig élő véletlen számmal azonosított tokent, amelyet elment a felhasználóhoz. A token azonosítóját belefűzi egy olyan URL-be, amelyet csak a felhasználó kap meg a megadott E-mail címére.

AZ URL PÉLDÁUL ÍGY NÉZHET KI:

http://admin.expandrive.io/reset/5fb6dfa00b3cb9a754c6bc1a075b822a032e48b3

Erre a linkre kattintva a felhasználó az "új jelszó beállítása" nevű felületre jut. Ha a rendszer nem talált felhasználót az elküldött E-mail cím alapján úgy a hiba jelzésével újra betölti az elfelejtett jelszó felületet. Minden új jelszó kérést - legyen az sikeres, vagy sikertelen - naplózni kell, hogy a monitoring rendszer képes legyen szükség esetén értesíteni az illetékseket a felület visszaélésszerű használatáról.



Biztonság

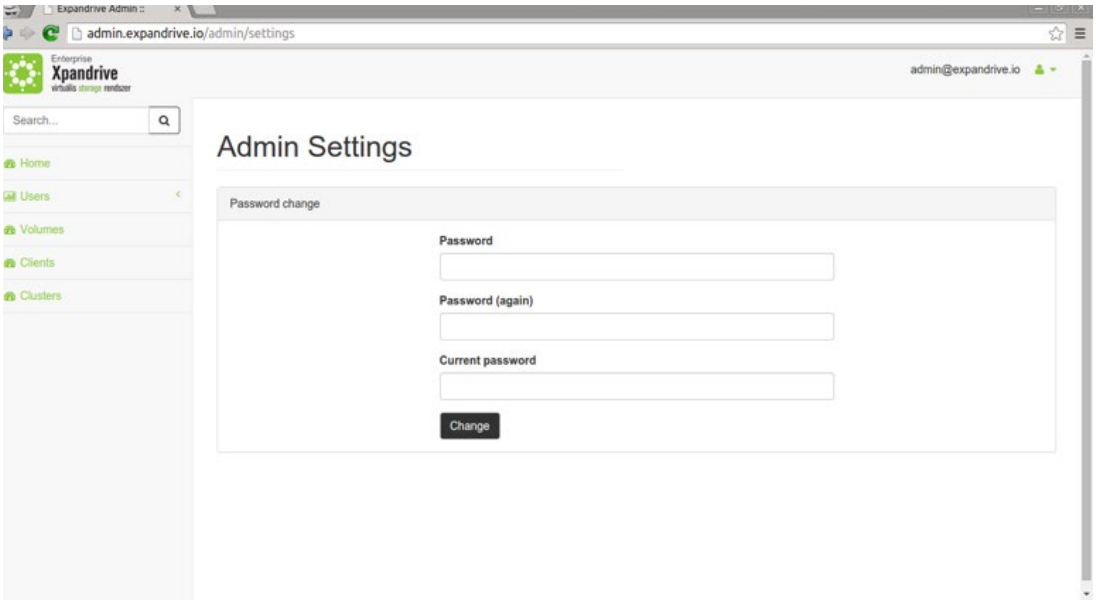
A használt folyamat miatt ennek a lehetősége igen csekély, hiszen a felhasználó e-mail fiókjának elfoglalása nélkül egy rosszindulatú felhasználó nem képes a jelszavak megváltoztatására a megcélzott felhasználó akarata nélkül.

Komoly biztonsági problémát jelent a CSRF, vagyis a Cross site request forgery. Egy sikeresen végrehajtott CSRF támadás ugyanis ezen felület esetén rávehet az alkalmazást, hogy a felhasználó nevében kérjen jelszó-visszaállító tokent.
Ez főleg akkor nagyon veszélyes, ha a felhasználók valamilyen web alapú e-mail szolgáltatást használnak, és abban megjelenik egy XSS, vagyis Cross-Site Scripting hiba. Képzeljük el a következő lehetőséget:

Egy felhasználó web alapú e-mail klienst használ. Ebben a kliensben létezik egy sebezhetőség, ami miatt a kapott e-mailek megfelelő formázása esetén tetszőlegesen formázott HTML jeleníthető meg, illetve tetszőleges javascript kód futtatható a felhasználó böngészőjében. Ez a kód képes lehet arra, hogy elolvassa a felhasználó leveleit, így a jelszó tokeneket is.

A KÖVETKEZŐ KÉPERNYŐLEÍRÁS AZ ÚJ JELSZÓ BEÁLLÍTÁSÁST RÉSZLETEZI.

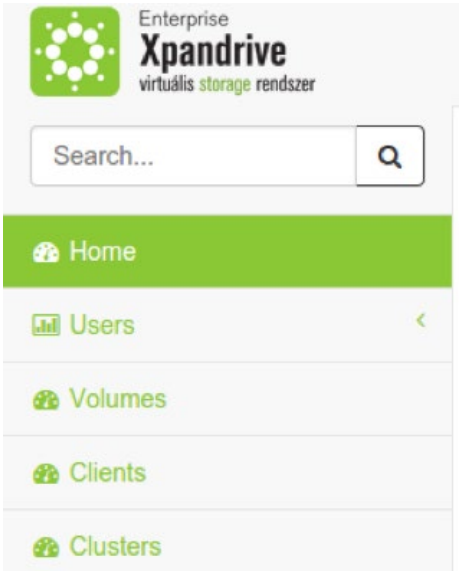
ÚJ JELSZÓ BEÁLLÍTÁSA



A Reset Password felületen adhatjuk meg új jelszavunkat, melyet szokásosan kétszer kell megadnunk az elírás elkerülése végett. Az itt megadott új jelszóra ugyanazok a megkötések vonatkoznak, mint az új felhasználó felvételénél megadott jelszavakra. Ezeket a feltételeket célszerű kliens oldalon ellenőrizni, és azokat a felületen jól olvashatóan, előre fel kell tüntetni.

Az "OK" gombra nyomva a frontend elküldi a backend-nek a kitöltött űrlap adatokat egy POST kérésben, ahol a kontroller összeveti a két paramétert, és ha azok egyeznek, akkor a jelszót behelyettesíti a felhasználó korábban megadott jelszavának a helyére, és kitörli a korábban legenerált hash-tokent, így az többé nem használható. Ezek után a kontroller betölti a bejelentkező nézetet, és visszajelez, hogy az új jelszó beállítása sikeres volt.

Amennyiben a két megadott jelszó nem egyezik, úgy a kontroller újra meghívja a korábbi felületet, és nem törli a legenerált hash-tokent, így a felhasználó többször is próbálkozhat, a token érvényessége csak időhöz kötött. Amennyiben 1 napig nem történik jelszó-módosítási kisérlet, vagy nem sikerül megfelelő jelszót beállítani, úgy automatikusan törlődik a hash-token. Ez csökkenti annak valószínűségét, hogy a tokenhez illetéktelenek férjenek hozzá. A sikeres jelszóváltoztatás naplózásra kerül, hogy a monitoring rendszer szükség esetén.



Menüpontok

Ha a felhasználó sikeresen bejelentkezett, akkor a képernyő bal oldalán található oldalsávban látja a menüt, amin keresztül elérhetőek a kezelt objektumokon végezhető műveletek.

Minden egyes menüpont egy-egy összegző oldalra vezet, ahol az adott objektum típus egyes elemei láthatóak az alapvető adataikat megjelenítve. Az egyes típusokra történő kattintással juthatunk az adott objektum típus kiválasztott elemének kifejtő nézetére.

Alább a menüpontok kibontása következik.

BEJELENTKEZETT FŐOLDAL

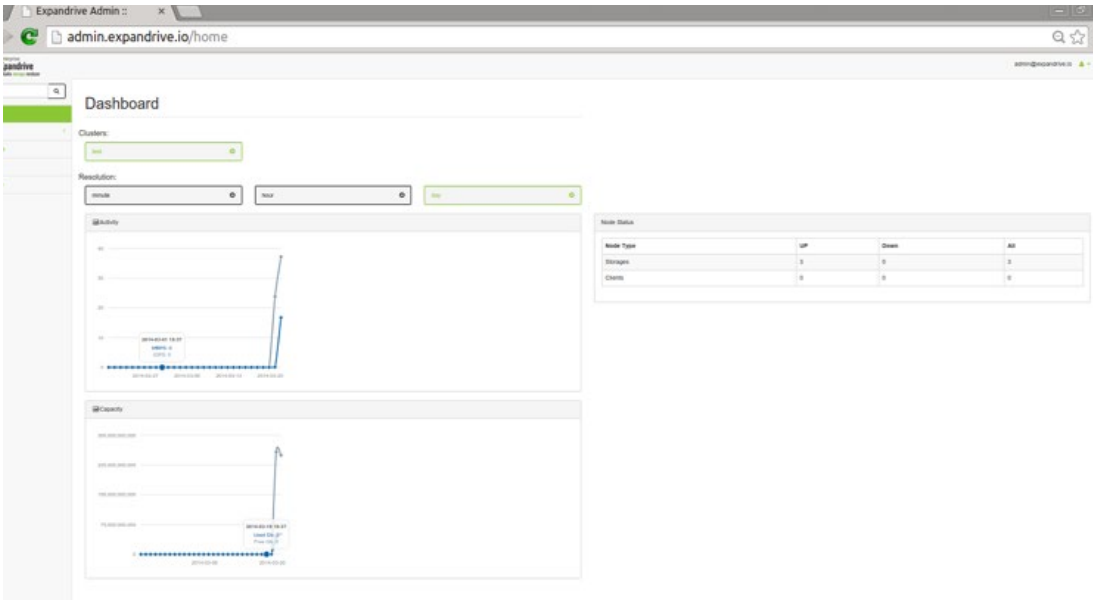
Sikeres bejelentkezést követően az általános összegző nézetet láthatjuk fülönként szeparálva az egyes clusterekhez tartozó információkkal. Minden egyes fülön láthatjuk az adott cluster aktivitását és az adott clusteren elérhető szabad hely méretét. Mindkét adat egy-egy grafikon által van vizalizálva.

A betöltött nézet minden egyes fülén a jobb oldali részén láthatjuk, hogy az adott clusteren az egyes Node típusok milyen mennyiségben, illetve hogy az adott mennyiség közül hány node működik, illetve nem működik.

Grafikonok

Minden egyes fül alatt, tehát minden egyes tárklaszterhez tartozik két grafikon. Az egyik a kapacitást, a másik a forgalmat mutatja. Az aktivitás grafikonról olvasható a használt sávszélesség (MByte/másodperc), illetve az elvégzett IO műveletek száma másodpercenként (IOPS). A grafikon egy vonalgrafikon, ami mindkét adatvonalat egyszerre tartalmazza. A függőleges tengelyt a bal- és a jobb oldalon is felcímkézzük. A bal oldalon látható a MB/s skála a jobb oldalon az IOPs skála. Az aktivitásgrafikon vízszintes tengelyén az időt jeleníti meg.

A webes alkalmazás szempontjából a klaszteren, nem pedig a köteteken elérhető hely a fontos, ezért ez a grafikon a klaszter gépein, a társzolgáltatás számára elérhető helyet mutatja.



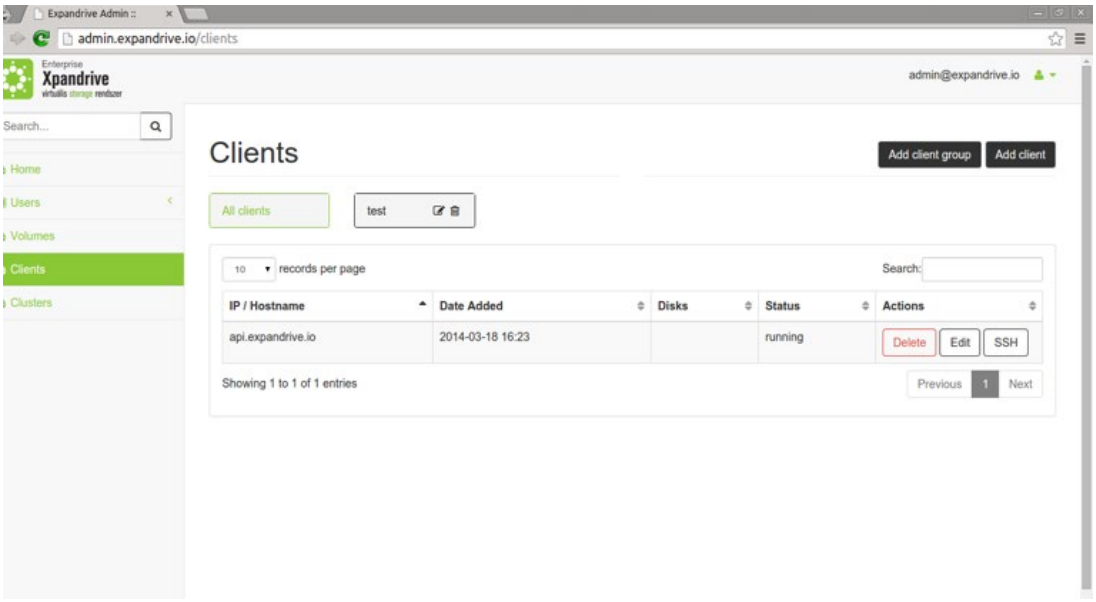
CSOMÓPONTOK ÁLLAPOTA

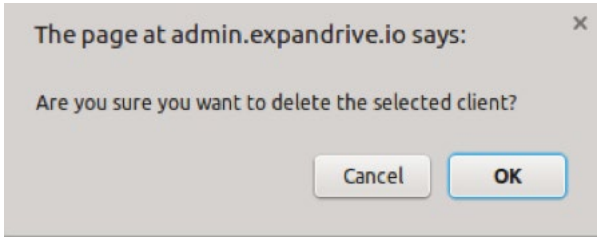
A csomópontok működését Zabbix felügyeli. A webes adminisztrációs felület a Zabbix szerverhez csatlakozik, hogy a csomópontok állapotát figyelje. A tár- és kliens csomópontok működését a Zabbix api-n keresztül mutatja ez a táblázat.
A táblázatban látható a csomópontok száma, illetve hogy pillanatnyilag ebből mennyi működik, és mennyi nem működik.

Az adatok frissítése

A táblázat és a grafikonok tartalmát 30 másodpercenként AJAX hívással frissíti a felület.

Kliensek (Clients)



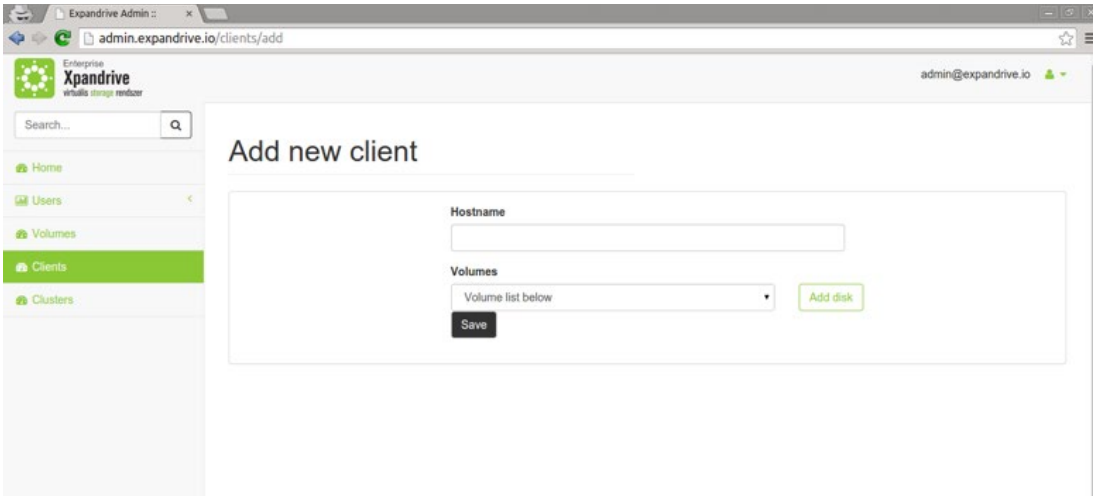


A kliens csomópontok összegző nézetében láthatjuk az egyes kliensekhez kapcsolódó információkat, többek közt hogy mely IP címről, vagy Host-névvel lett felvéve, láthatjuk a hozzádás dátumát, illetve, hogy az adott klienshez mely kötetek vannak hozzárendelve.

Legutolsó sorban láthatjuk azt a gombot, amellyel az adott klienst eltávolíthatjuk a rendszerből. A rendszer minden egyes elem-eltávolításkor megjelenít egy megerősítő ablakot, amelynél még eldönthetjük, hogy valóban az adott elemet szeretnénk-e eltávolítani a rendszerből. Az “OK” gomb minden esetben végrehajtja az eltávolítást, míg a “Cancel” gombbal megszakíthatjuk a folyamatot, mielőtt az még bármilyen vátlozást végrehajtott volna a rendszerben.

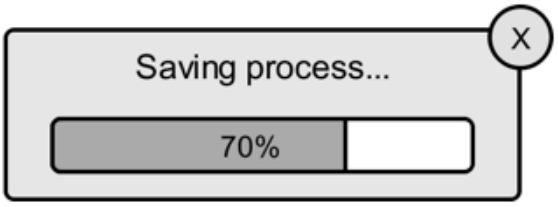
Az adott sor bármely más mezőjére kattintva juthatunk el a szerkesztő-felületre, ahol a kliens korábban megadott adatait tudjuk módosítani. A jobb felső sarokban láthatjuk az “Add new” gombot, mellyel új Klienst adhatunk hozzá a rendszerhez.

ÚJ KLIENS HOZZÁADÁSA



Új kliens hozzáadása esetén mindenképp ki kell töltenünk az IP cím, vagy a Hostname mezőt, és felvennünk, hogy a kliens mely kötetet használja majd. A kötetek hozzáadása a következőképp történik:

1. A legördülő listából kiválasztjuk a már felvett kötetet.
2. A (+) gombra kattintva az legördülő menü értékét JavaScript segítségével beszúrjuk egy táblázatba, mely minden egyes hozzáadás során frissül.
3. Az így kiválasztott adatok eltárolódnak egy rejtett form elemben is, amit mentésnél tömb formában küldünk tovább a szervernek.



A "Save" gombot megnyomva az űrlap összes elemét elküldjük a szervernek, amely feldolgozza az elküldött adatokat, és elmenti az adatbázis táblákba a őket, majd újra megjeleníti kliensek listáját, immár mutatva a frissen elmentett elemet is.

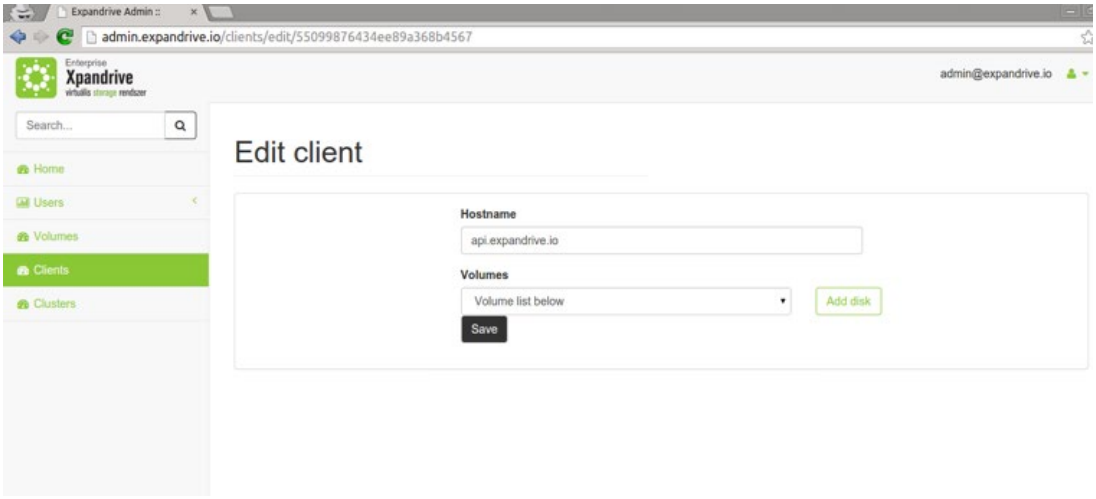
A mentés, illetve az új kliens felvétele akár néhány percig is eltarthat, ezért a mentés után egy várakoztató képernyő jelenik meg, ami tájékoztat a folyamat előrehaladásáról. A kliens lista csak sikeres felvétel után jelenik meg.
Hiba esetén megjelenik a futtatott parancsok kimenete, amiből következetni lehet arra, hogy hol történt a hiba.

A háttérben ilyenkor a következő történik:

1. A szerver elmenti az adatbázisba az új klienst minden adatával együtt.
2. A chef szerveren a chef API-n keresztül létrejön egy új node, és bekerül a "client" role-ba.
3. Elindul egy folyamat, ami ssh segítségével bejelentkezik a gépre, és feltelepíti a chef klienst. Telepítés után rögtön le is futtatja.
4. Minden parancs kimenete naplózódik.
5. Akár sikeres a telepítés, akár nem, a chef-client service futni fog a gépen.

Sikertelen telepítés esetén a kliens a rendszerben marad, és a rendszer nem próbálja meg eltávolítani, vagy visszacsinálni az elvégzett dolgokat. Ilyenkor emberi beavatkozás szükséges, és meg kell próbálni kézzel helyrehozni a hibát. Erre a célra elérhető egy terminál képernyő, ami a szerver ssh kulcsával képes elérni a klienst (amennyiben ez a kliensen helyesen be van állítva).

KIVÁLASZTOTT KLIENS SZERKESZTÉSE



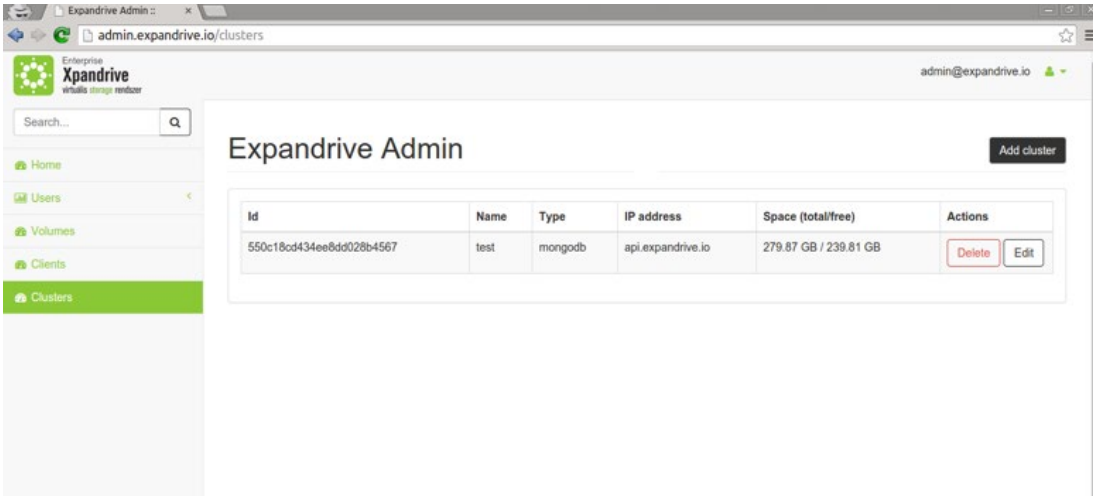
Kliens szerkesztés esetén az első mező már a korábban kitöltött paramétert tartalmazza, melyet ezúttal is átszerkeszthetünk, kijavíthatunk. Alatta láthatjuk a már korábban felvett köteteket, melyekhez a már korábbi módon felvehetünk újabbakat is, illetve törölhetjük is a korábbiakat.

A felvétel a korábbiak szerint zajlik:

1. A legördülő listából kiválasztjuk a már felvett Volume-ot.
2. A (+) gombra kattintva az legördülő menü értékét JavaScript segítségével beszúrjuk egy táblázatba, mely minden egyes hozzáadás során frissül.
3. Az így kiválasztott adatok eltárolódnak egy rejtett form elemben is, amit mentésnél tömb formában küldünk tovább a szervernek.

Az "Update" gombot megnyomva az űrlap összes elemét elküldjük a szervernek, amely feldolgozza az elküldött POST kérést, elmenti az adatbázis táblákba a tartalmakat, majd újra meghívja a kliensek listáját, immár a friss adatokat megjelenítve.
A szerver felveszi a kapcsolatot a chef szerverrel is, és beállítja az új köteteket. A kliens csomópontnál látható kötetek ezért nem feltétlenül jelentik azt, hogy a kötet valóban használható az adott csomóponton.

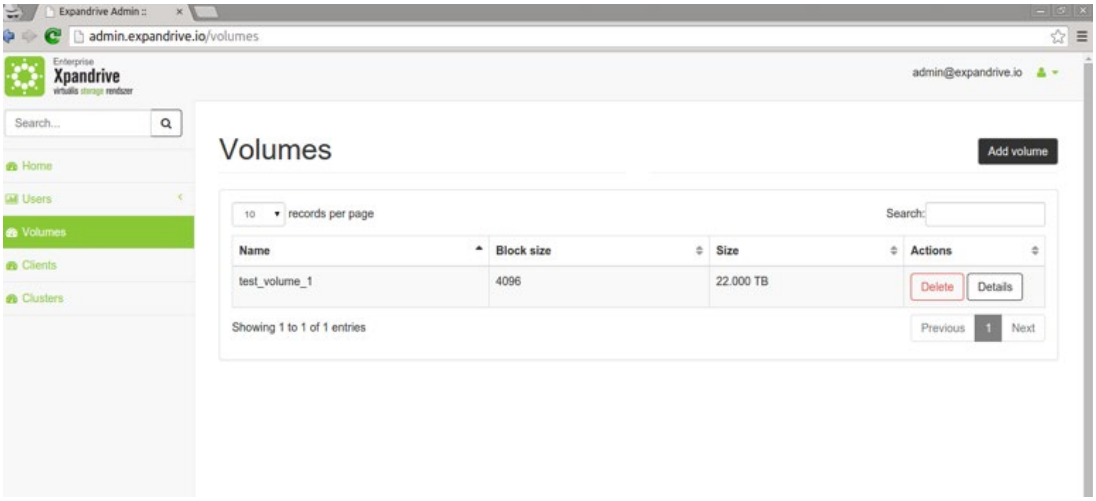
TÁRKLASZTER - MONGODB



A MongoDB típusú kifejtő nézetnél láthatjuk a Storage nevét, melyet módosíthatunk is.

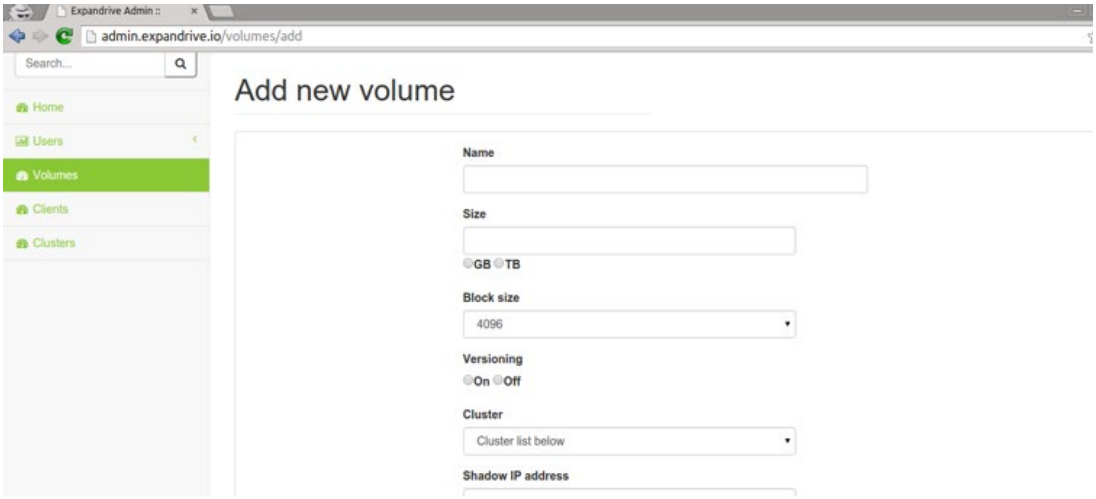
A "Save" gomb megnyomásával a módosított adatokat egy POST kérésben továbbküldjük a szervernek, a kontroller az adatok ellenőrzése és feldolgozása után meghívja a listázó nézetet, amennyiben nem történt hiba. Hiba esetén a kontroller újra meghívja a kifejtő nézetet, jelezve hogy milyen hiba történt, és hogy hol kell azt kijavítanunk, ha az a saját hibánkból történt.

KÖTETEK



A kötet listázó nézetében láthatjuk az egyes kötetekhoz kapcsolódó információkat, többek közt a Volume nevét, a bokkméretet, és a kötet méretét. Az adott sor "details" gombjára kattintva, vagy az adott sor juthatunk el a szerkesztő-felületre, ahol a Volume korábban megadott adatait tudjuk módosítani. Új Volume-ot a jobb felső sarokban található "Add new" gombra kattintva tudunk felvenni.

KÖTET LÉTREHOZÁSA



Új kötet felvételénél meg kell adnunk, hogy mi legyen a kötet neve. A kötet ezzel a névvel lesz felvéve a kliensek /dev/ könyvtárába, ezért érdemes olyan neveket használni, amelyek egyrészt már létező blokkos eszközökkel nem ütköznek.

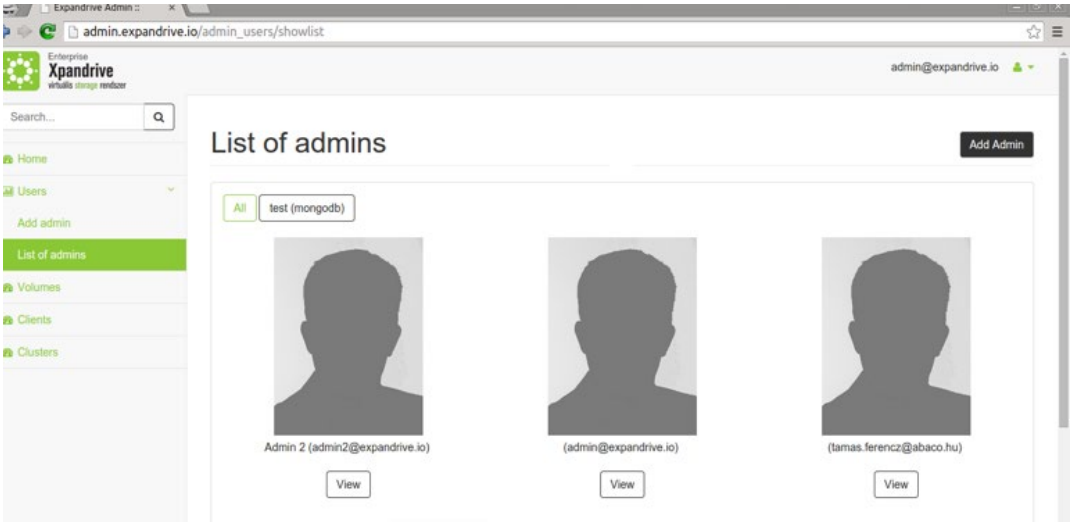
Ezt követően meg kell adnunk, hogy az adott Volume mekkora méretűnek szeretnénk beállítani, illetve hogy ezt a méretet GB-ban, vagy TB-ban szeretnénk megkapni. Mivel minden tárklaszter fel van készítve arra, hogy csak a ténylegesen használt blokkokat tárolja, ezért a művelet elvégzése a tárhely szerint O(1) időben és ugyanekkora tárhelyben végrehajtható.

Ez a fájlremdszerek esetében nem igaz, a nagyon nagy fájlrendszerek létrehozása gyakran nagyon sok időbe, illetve jelentős tárhelybe kerülhet. Ezek miatt nyugodtan meg lehet adni egy akkora tárméretet, ami “nagyon sokáig elég”, de ezzel együtt vigyázni kell arra is, hogy túl nagy tárterületek létrehozásával ne pazaroljunk feleslegesen a fizikai tárhelyből. A blokkméret befolyásolja, hogy az adatok mekkora adagokban közlekednek a tár és a kernel között. Ha nincs különösebb okunk, válasszuk a 4096 byte-os blokkméretet. A verziókövetés egy kétállsú kapcsoló, mellyel beállíthatjuk, hogy az adott kötet tartalmára állítunk-e be verziókövetést. A verziókövetés minden blokk minden verzióját megőrzi. Ez intenzíven írt kötetek esetén nagyon sok helyet felemészthet, cserébe viszont visszamenőleg tetszőleges időpontra “visszatekerhetjük” a kötetet, így nagyon gyorsan lehet véletlenül letörölt, felülírt, vagy más módon megsérült adatokat visszanyerni.

Az árnyéktár, vagy shadow storage hagyományos értelemben egy olyan tármegoldás, ami képes működés közben mentéseket készíteni egy blokkos eszközről. Az ExpanDrive megoldás már nyújt egy hasonló, verziókövetésre épülő megoldást. Más, külső megoldások könnyeb csatlakoztatásának érdekében azonban az ExpanDrive kötetek képesek egy egyszerű TCP csatornánk keresztül átadni az elvégzett IO műveleteket. A shadow storage nálunk egy külső szolgáltatás, ami TCP-n keresztül, egy általunk megadott protokollal érhető el. Egy ilyen szolgáltatás célja lehet biztonsági mentések készítése (nem verziókövetett rendszer esetében), vagy behatolásérzékelés, vírusellenőrzés, stb.

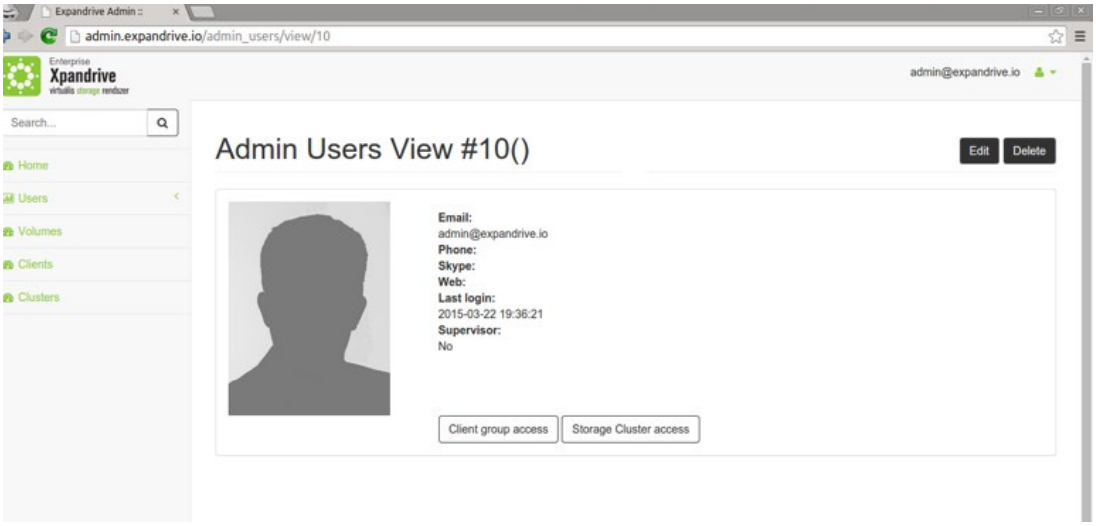
A “Create” gombra nyomva egy POST kérésben továbbküldjük az adatokat a szervernek, ahol a kontroller osztály feldolgozza, és ellenőrzi az elküldött adatokat. Amennyiben nem talál hibát, a kontroller meghívja a Volumes listát, ahol már láthatjuk az újonnan az általunk felvett elemet is. Amennyiben hibát észlelne a kontroller osztály, úgy újra meghívja a létrehozó nézetet, ahol jelzi a hibát, és hogy mely adat kitöltésénél hibáztunk. Amennyiben rendszerhiba történt volna, vagy bármilyen korlátozásba ütközne a folyamat,
A shadow storage mutatja, hogy van-e külső árnyéktár csatlakoztatva, és az milyen IP címen és portszámon érhető el.

FELHASZNÁLÓK



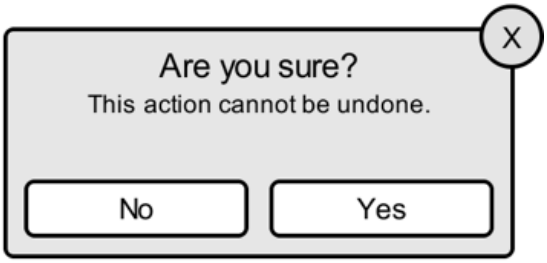
A felhasználók listázó nézetében tabokra bontva láthatjuk az egyes tárklaszterekhez rendelt felhasználókat névvel, és avatárral ellátva. A “névre”, vagy a “Details” gombra kattintva eljuthatunk a felhasználó kifejtő nézetére, míg a jobb felső sarokban lévő “New” gombra kattintva új felhasználót adhatunk hozz a rendszerhez.

FELHASZNÁLÓI ADATLAP



A felhasználó kifejtő nézetében láthatjuk a kiválasztott felhasználóhoz tartozó azon adatokat, amelyeket korábban kitöltöttünk. Jobb oldalon láthatjuk a felhasználó profilképét, míg a bal oldalon láthatjuk az E-mail címét, a megadott telefonos, és Skype elérhetőségét, valamint ha ki van töltve, akkor a weboldalát is. Ezen felül láthatjuk, hogy az adott felhasználó mikor lépett be utoljára, és hogy mely tárklaszterekhez van hozzáférése.

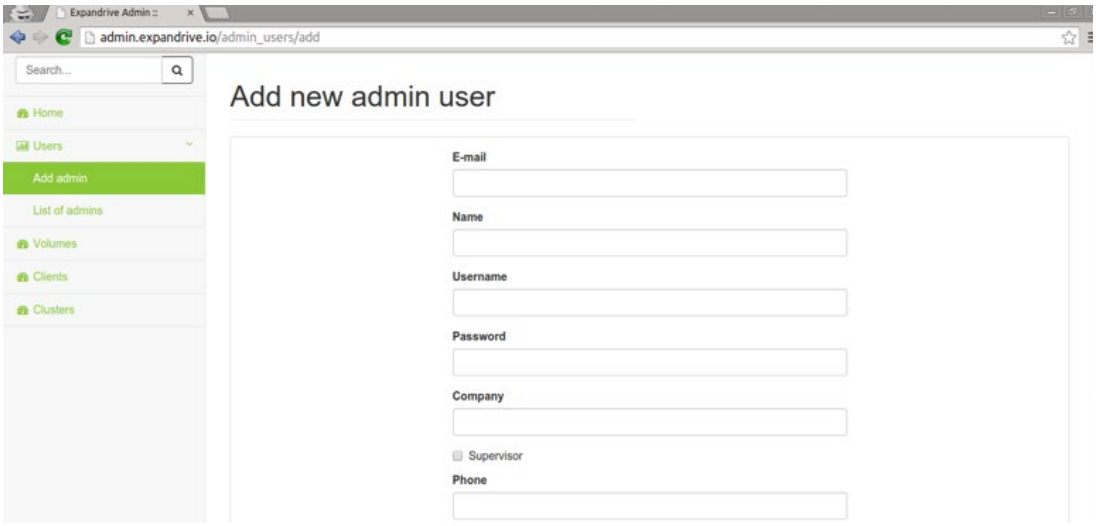
Az “Edit” gombra nyomva a frontend elküldi a backendnek a felhasználó egyéni azonosítóját, majd a kontroller az alapján előkeresi a felhasználó adatait, összegyűjti egy tömbbe, majd a tömböt átadva meghívja az adott felhasználó adatait tartalmazó szerkesztő-nézetet.

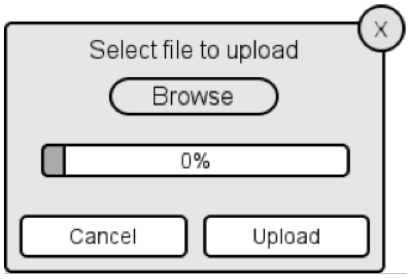


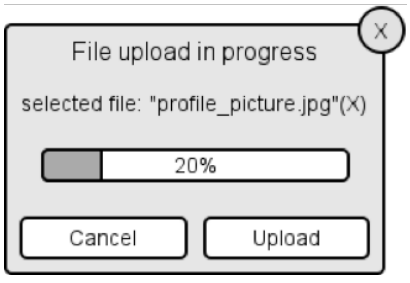
A “Remove” gombra nyomva először megjelenik egy megerősítést kérő dialógus ablak, ahol a “No”-ra nyomva megszakíthatjuk a folyamatot. Ha ebben a dialógus ablakban OK-re nyomunk úgy a Frontend elküldi a felhasználó egyéni azonosítóját a backend-nek, ahol az adatbázisból az összes adott felhasználóhoz tartozó adat eltávolításra kerül.

A kontroller a törlés után meghívja a felhasználókat listázó nézetet.

FELHASZNÁLÓ LÉTREHOZÁSA







Új felhasználó esetán három blokkba rendezve láthatjuk az egyes egymáshoz tartozó kitöltendő információkat. Bal szélen láthatjuk a felhasználó profilképét, melyet az “Upload” gombra kattintva megjelenő dialógus ablakban a “Browse” gombbal tudunk kiválasztani. A kiválasztás után láthatjuk az általunk megjelölt fájl nevét, és a mellette lévő “(X)”-re kattintva eltávolíthatjuk, és kereshetünk másik képet.

Ezt követően dialógus ablakon belül is az upload gombra kattintva feltölteni.
Az upload gombot megyomva a feltöltési folyamat elkezdődik, melyet egy folyamatjelző ábrán is láthatunk.

A középső blokkban találhatóak a felhasználó alapértelmezett adatai, melyek közül a csillaggal megjeleölt részeket kötelezően ki kell tölteni, azok hiányában a save gombra nyomva hibaüzenetet kapunk. Az alap adatokon felül itt tudjuk kiválasztani, hogy elsődlegesen az adott felhasználó melyik tárklaszterhez férhet hozzá.

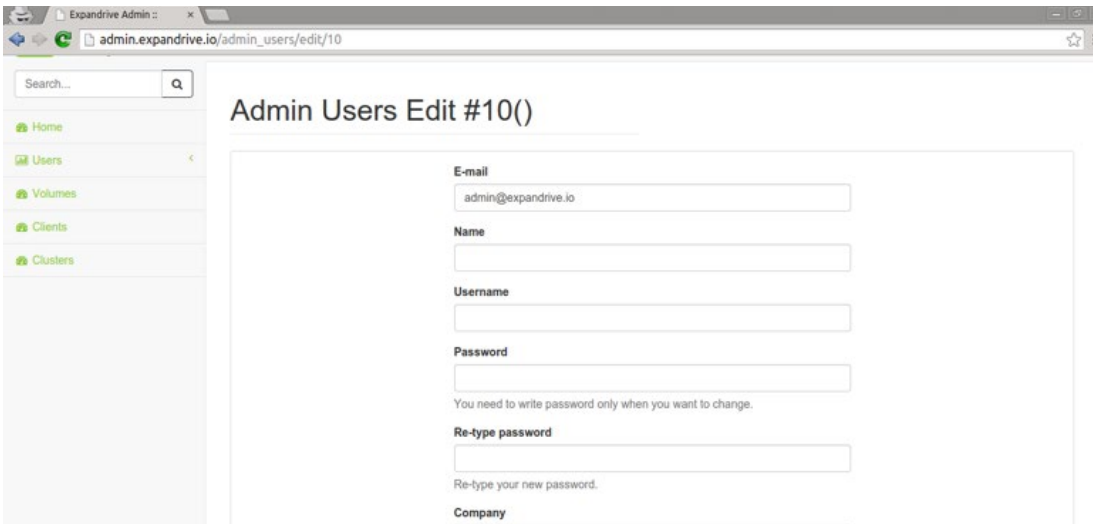
A jobb blokkban tudunk a felhasználóhoz jelszót megadni. Ezt a lépést biztonsági okokból, az elírást megelőzendő, kétszer kell megtennünk. Ha bármelyik mező tartalma eltér egymástól, azt adott mező elszínezésével láthatjuk. A Save gomb megnyomásával elmenthetjük a felhasználót.

A Save gomb megnyomásával először lefut egy kliens oldali ellenőrzés, ahol egy javascript kód ellenőrzi, hogy az összes kötelezőnek jelölt űrlap elem tartalmaz-e adatot, illetve hogy az az adat formailag megfelelő-e. Ha az ellenőrzés sikertelen, azon űrlap-mezők, ahol helytelenül, vagy egyáltalán nem volt kitöltve adat, pirosra váltanak, ezzel is jelezve, hogy ott még javítandó, kitöltendő adatokat vár a rendszer. Ha az ellenőrzés sikeres, úgy a javascript kód kezdeményezi az űrlap elküldését a szervernek, mely egy POST kérésben fog küldésre kerülni. A Backend ezek után még egyszer érvényesíti a tartalmakat. Ha mindent rendbe talál, akkor a felhasználó adatait elmenti az adatbázisba, és lekérdezi az így létrejött felhasználó egyéni azonosítóját.

Ha az adott felhasználóhoz töltöttek fel profilképet is, úgy a rendszer az adott profilképet átméretezi egy 200px széles változatra, és átnevezi az adott felhasználó egyéni azonosítójának a nevére (tehát a fájlnév például így fog kinézni: 12.jpg), majd ezt a képet elmenti a rendszer a felhasználói profilképeket tároló mappába.

A képek betöltési menete mindig úgy zajlik, hogy megvizsgáljuk, hogy az adott felhasználó egyéni azonosítójával létezik-e fájl a megadott könyvtárban. Ha létezik, akkor megjelenítésre kerül, ha nem, akkor az alapértelmezett profilkép fog megjelenni.

FELHASZNÁLÓ SZERKESZTÉSE



A felhasználó szerkesztőnézete hasonló módon épül fel, mint az adott típust létrehozó felület. Bal oldalon láthatjuk a feltöltött profilképet, középen a kitöltött alapértelmezett adatokat. Illetve jobb oldalt láthatjuk a jelszó megváltoztató mező. Minden adatot módosíthat mind a felhasználó, mind az ő profilját létrehozó adminisztrátor.

Jelszó módosításakor, akárcsak a unix szerű rendszereknél, meg kell adnunk az aktuális jelszót, majd ugyanúgy, mint létrehozásnál, kétszer kell megadnunk az új jelszót, melynél ha bármelyik mezőt elütjük a másikhoz képest, az adott mezőnél jelezzük a problémát. Az Update gombra kattintva a módosított adatokat egy POST kérésben elküldjük a szervernek, ahol a kontroller osztály feldolgozza és ellenőrzi a tömb tartalmát, hogy az megfelel-e az elvárt formáknak. Amennyiben nem talál hibát, abban az esetben a friss adatokat elmenti az adatbázisba, majd betölti a felhasználók listanézetét.

BLOKKOS ESZKÖZÖK KEZELÉSE

A szoftverrendszer dinamikusan, futás időben képes létrehozni és törölni blokkos eszközöket. A blokkos eszközö nem feltétlenül azon a fizikai (vagy viztuális) gépen vannak jelen, ahol az adat fizikailag tárolódik. Jelen megoldás egyik fontos eleme pontosan az, hogy az adattárolást az arra speciálisan felkészített felhőben végezzük, amelynek elemei a tárcsomópontok.

Egy ilyen felhő sok blokkos eszköz kiszolgálására képes, és az eszközök maguk is több folyamatnak tudnak szolgáltatást nyújtani több szálon. Az egyes blokkos eszközök létrehozását és eltávolítását, illetve életciklusuk további állomásain való átsegítését egy parancssori eszköz segíti.

Az eszköz neve bdtun. Az alábbi néhány sor a parancs kimenetét mutatja a --help kapcsolóval meghívva:

```
Usage: bdtun [OPTION...]
Manage bdtun device pairs

 Commands
  -c, --create            create a tunnel (with -bns)
  -i, --info              get info on a tunnel (with -n)
  -l, --list              list existing tunnels
  -r, --remove            remove a tunnel (with -n)
  -z, --resize            resize the block device (with -ns)

 Options
  -b, --block-size=BLOCKSIZE device block size in bytes
  -n, --name=NAME             name of tunnel
   --req-discard          support REQ_DISCARD
   --req-flush            support REQ_FLUSH
   --req-fua              support REQ_FUA
   --req-secure           support secure discard (REQ_SAFE)
  -s, --size=SIZE         block device size in bytes

 Informational options
  -?, --help              Give this help list
   --usage                Give a short usage message
  -V, --version           Print program version
```

A bdtun parancssori eszköz csupán az eszközt hozza létre a kernelben, illetve a /dev/ könyvtár alatti eszközfájlokat hozza létre. Az IO kérések kiszolgálása már egy külön erre a célra létrehozott daemon folyamat feladata.

SKÁLÁZHATÓSÁG

Hasonlóan fontos tulajdonság, hogy a tárkapacitás, vagy teljesítmény bővítése gyorsan és egyszerűen elvégezhető újabb csomópontok felvételével a felhőbe. A mára népszerűvé vált NoSQL megoldások gyakran szinte automatikusan teljesítik ezt a követelményt.

A **MongoDB** egy strukturált dokumentumtár, ami képes bináris adatokat is jó hatásfokkal tárolni. Az aktuális szempontból egy fontos képessége az autosharding. Az autosharding működés lehetővé teszi, hogy a mongodb több csomópont között ossza szét a tárolt adatokat, és olvasás esetén csak egyetlen csomóponton történjen keresés / olvasás műveklet. Gyorsan meggyőződhetünk róla, hogy sok szál és véletlenszerű IO műveletek esetén a szálak által a rendszerre kifejtett nyomás egyenletesen eloszlik az egyes csomópontok között, így egy újabb csomópont felvétele arányosan és lineárisan csökkenti a rendszerbeli nyomást.

KERNEL DRIVER

A kenel driver egy vékony réteg, ami kapcsolatot tart a felhasználók folyamatai, és a háttértárral kommunikáló daemon folyamat között. Ez úgy történik, hogy a driver létrehoz egy blokkos, és egy karakteres eszközt. A blokkos eszköz egy hagyományos blokkos eszköznek látszik, és bármilyen szokásos művelet elvégezhető vele. A karakteres eszközön keresztül jutnak az IO kérések a háttértár szolgáltatás felé kommunikáló daemonhoz. A daemon egyenként olvassa az IO kéréseket a karakteres eszközről, és szolgálja ki azokat. A kernel driver egyszerre több blokkos-karakteres eszközpárt (alagutat, tunnelt) képes kezelni.

DKMS

A DKMS, azaz Dynamic Kernel Module Support egy keretrendszer, ami segíti azoknak a kernel moduloknak a telepítését és kezelését, amiknek a forrása nincs benne a kernel forrásfájában. Többek között azt biztosítja, hogy ezek a kernel modulok újraépülnek akkor, ha az operációs rendszerre új kernelt telepítenek. A DMKS továbbá megkíméli a felhasználót a fáradságos kézi fordítástól és telepítéstől.

Debian package

A kernel modul (a parancssoros eszközökkel együtt) közvetlenül .deb csomagból telepíthető. A csomag gondoskodik arról, hogy a modul DKMS-sel helyesen együttműködjön.

ProcFS/SysFS paraméterek

A bdtun kernel modul igyekszik a hagyományos blokkos eszközök minél több funkcióját biztosítani. Az alábbi fájlok többé-kevésbé szabványos (de lagalábbis gyakran használt) elemek, amik információt adnak az operációs rendszerről, és érdekesek a bdtun szempontjából:

- /proc/devices - karakteres és blokkos eszközöket tartalmazza. Itt megjelennek a bdtun kernel modul által létrehozott blokkos és karakteres eszközök is.
- /proc/diskstats - a bdtun eszköz jelent a kernel statisztikai funkciói felé, így a használati adatok megjelennek ebben a fájlban, és az iostat parancs is képes helyes statisztikákkal szolgálni.
- /proc/modules - a bdtun modul megjelenik a betöltött kernelmodulok listájában
- /proc/partitions - a blokkos eszközökön lévő partíciók listája, így a bdtun eszközökön lévőket is mutatja.

Modul paraméterek

A kernel modul nem rendelkezik paraméterekkel, az összes működési beállítás futásidőben történik.

ENTERPRISE
GROUP

MEGOLDÁSOK ÉS SZOLGÁLTATÁSOK

ICT
ÜZLETÁG

Komplex IT megoldások, IP telefónia
és csoportmunkát támogató Egységes
Kommunikációs megoldások (UCC).

PLM
ÜZLETÁG

CAD/CAM megoldások és termékciklus
menedzsment (PLM) a tervezéstől
a megvalósításig.

eHEALTH
ÜZLETÁG

Technológia a gyógyítás szolgálatában
– új távlatok az egészségügyi informatikában.

CONSULTING
ÜZLETÁG

Iparág specifikus SAP bevezetés és tanácsadás
– versenyképesség a legújabb technológiák
felhasználásával.

H-1138 Budapest, Váci út 117-119. • Tel.: +36 1 471 2400 • Fax: +36 1 471 2402