Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Irányítástechnika és Informatika Tanszék

Kiszolgálók üzemeltetése nagyvállalati környezetben és kapcsolódó szoftverek fejlesztése

Szakdolgozat

Készítette

Főglein Simon István

Konzulens

dr. Somogyi Péter

2024. május 8.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Főglein Simon István*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. május 8.

Főglein Simon István
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

1. fejezet

Bevezetés

Napjainkban az informatika és az internet életünk szerves részévé vált. A számos szolgáltatás folyamatos rendelkezésre állásának biztosítása és a megnövekedett forgalom kiszolgálása jó néhány új technológia kifejlesztését követelte meg.

Dolgozatomban elsősorban egy általános képet szeretnék adni arról, hogy milyen üzemeltetési kihívásokkal kell szembenéznünk, ha egy ilyen szolgáltatás működtetésébe vágjuk a fejszénket. Értekezésemben nem fogok kitérni bizonyos infrastrukturális hátterekre – mint például a kiszolgálók folyamatos energiaellátásának biztosítása –, ezeket adottnak fogom tekinteni, hiszen ezt egy adatközpontban bérelt hely esetén sem magunknak kell biztosítanunk. A következőkben sokkal inkább az informatikai lehetőségek tárgyalására fogom helyezni a hangsúlyt: hogyan tudunk hatékonyan üzemeltetni több kiszolgálót, milyen módon lehet biztosítani a szolgáltatásaink lehető legnagyobb rendelkezésre állását, és hogyan védhetjük meg adatainkat egy esetlegesen félresikerült rendszerfrissítést követően.

A dolgozatban érinteni fogom a jelenleg legelterjedtebb virtualizációs technológiákat, melyek főbb tulajdonságait röviden ismertetem, valamint össze is hasonlítom ezeket a megoldásokat a legfontosabb különbségekre kitérve.

Szerepet fog kapni továbbá a logikai kötetkezelés, ezen belül is a Linux kernelben elérhető LVM-implementáció. Ez a technológia nagyban megkönnyíti a háttértárak és partíciók kezelését üzemeltetési szempontból, melyet főként virtualizációt végző fizikai gépek esetében használhatunk ki, hiszen ilyen helyzetekben érdemes minden virtuális rendszernek külön partíciót létrehozni, amelyek kezelése (pl. egy esetleges bővítés során) a hagyományos particionálási megoldásokkal sokkal összetettebb feladat lenne.

Szót ejtek a monitoring megoldásokról is, melyek elengedhetetlenek ahhoz, hogy a rendszer üzemeltetését végző szakemberek pontos képet kapjanak az infrastruktúra ak-

tuális állapotáról, az esetleges korábbi problémákról. A monitorozás azért is fontos, mert ha egy hibát ezáltal sikerül idejekorán felismerni (például háttértárak esetén egy megfelelő határék beállításával időben értesülhetünk egy partíció megteléséről, és nem csak az írási hibákat tapasztaljuk), akkor elkerülhetőek a további, komolyabb hibák, amik akár a felhasználók számára is fennakadásokat okozhatnak. Az általam létrehozott tesztkörnyezetben is bemutatok egy ilyen monitoring megoldást, melynek segítségével az általam létrehozott infrastruktúra gépeit fogom folyamatosan ellenőrizni.

A tesztkörnyezet beállításában nagy szerep fog jutni a választott konfigurációmenedzsment szoftvernek, a Salt-nak. Ez arra fog lehetőséget biztosítani, hogy egyes konfigurációs fájlokat egyszerűen telepíthessünk több számítógépre is, valamint a keretrendszer leíró nyelvén meghatározott konfigurációleíró szoftver lehetővé teszi azt is, hogy ellenőrizzük egyes szolgáltatások (service) állapotát. Ez hasznunkra válhat például egy saját serviceszel érkező program telepítését követően, hiszen így a leíróban megadhatjuk a telepítés paramétereit, majd ezt követően egyből ellenőrizhetjük is, hogy a telepítés után sikeresen elindult-e az újonnan telepített szoftver.

A dolgozatban tárgyalt koncepciókat egy kisebb volumenű tesztrendszeren keresztül fogom bemutatni. Ennek a rendszernek a célja nem egy teljes vállalati környezet bemutatása, hiszen ehhez nagy mennyiségű hardverre, jelentős mértékű hardveres és szoftveres erőforrásokra lenne szükség, amelyek üzembe helyezése, összehangolása túlmutat a dolgozat keretein. Ehelyett sokkal inkább arra szeretnék rávilágítani, hogy milyen eszközök állnak rendelkezésre egy ilyen nagyszabású infrastruktúra sikeres üzemeltetésének elősegítéséhez. Gondoljunk csak arra, hogy egy 5-10 számítógépből álló rendszer esetén kivitelezhető, hogy a rendszergazdák egyesével telepítsék a havi frissítéseket, azonban egy több száz, vagy több ezer kiszolgálóból álló nagyvállalati környezetben nem lenne egy reális elvárás.

Az ilyen és ehhez hasonló kihívások megoldására fogok lehetőségeket mutatni a 3. fejezetben. Szó lesz a gépek távoli kezeléséről, folyamatos karbantartásukról, automatikus biztonsági javításokról (patchek) való értesülésről, ezek telepítéséről. Tárgyalni fogom továbbá a rendszert alkotó eszközök monitorozását, metrikák gyűjtését is, továbbá szó lesz az egyre szélesebb körben elterjedő konténerizációs technológiákról, ezek használatáról vállalati környezetekben. Bemutatom azt is, hogy a megfelelő eszközökkel milyen gyorsan hozhatunk létre konténereket, és mennyire hatékonyan kezelhetjük őket akár egy böngészőből is. Fontos megjegyezni, hogy az itt említett technológiák kisebb környezetekben is használhatóak, azonban néhány esetben az ilyen rendszerek használata kevesebb előnyt

nyújt, mint amennyi munkát telepítésük és karbantartásuk igényel, így érdemes felmérni az informatikai rendszerrel szemben támasztott elvárásainkat, és ennek megfelelően dönteni a szükséges technológiai komponensekről.

A 4. fejezetben fogom ismertetni az általam készített tesztkörnyezetet, ennek felépítését, a tervezési döntéseket, komponenseit, valamint az ezzel kapcsolatos munkáim során felmerült nehézségeket, tapasztalatokat. Ebben a fejezetben a korábban tárgyalt technológiák közül általam választott megoldásokat fogom részletesebben ismertetni.

Végül a dolgozat utolsó fejezetében értékelni fogom az elért eredményeket, valamint röviden összefoglalom a projekt továbbfejlesztési lehetőségeit.

2. fejezet

Nagyvállalati környezetek ismertetése

A vállalatok egyre nagyobb hangsúlyt fektetnek az informatikai rendszereik fejlesztésére és karbantartására, üzemeltetésére. Az ezek által nyújtott szolgáltatások sok esetben jelentős könnyebbséget jelentenek egyes üzleti folyamatokban, és a megfelelően automatizált munkafolyamatok csökkentik az egyes munkavállalók által elvégzendő manuális feladatokat, és adott esetben az ügyfelek számára is segítséget nyújthatnak. Fontos tisztában lenni azonban azzal, hogy ezek a megoldások csak akkor működnek jól a mindennapi használat során, ha sikerül biztosítani a megfelelő rendelkezésre állást, tehát egy – a vállalat munkavállalói által a munkához nélkülözhetetlen – szolgáltatásnak munkaidőben folyamatosan elérhetőnek kell lennie. Előfordulhatnak olyan igények is, amik miatt bizonyos, a vállalat működésének szempontjából nélkülözhetetlen szolgáltatásoknak folyamatosan elérhetőnek kell lenniük, mert a működésük nem munkaidőhöz kötött (ilyen lehet például a szervezet weboldala, illetve levelezőszervere). Emellett a fent említett informatikai rendszerek karbantartásának is sokszor észrevehetetlennek kell lennie, azaz egy esetleges frissítés nem hátráltathatja a munkavégzést és nem csökkentheti a rendelkezésre állást.

Mivel az ilyen rendszerek általában nagy méretűek és összetettek, a tervezésük és az üzemeltetésük is nagy szakértelmet igényel. Éles környezetek tervezése során előtérbe kell helyezni a skálázhatóságot, hogy az adott rendszer esetleges jövőbeli bővítése során legyen lehetőség az elérhető erőforrásokat növelni a szükséges mértékben.

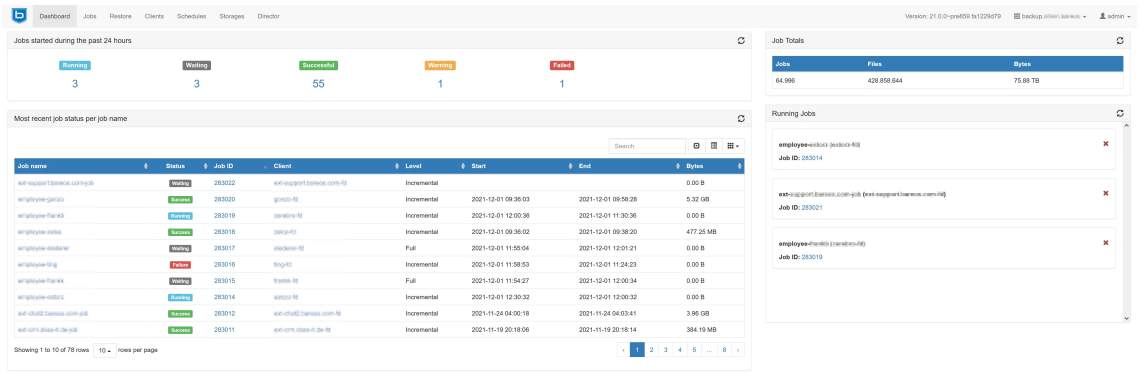
2.1. Adatvédelem/Biztonsági mentések/Biztonság

Egy másik fontos szempont a nagyvállalati rendszerek üzemeltetése – és általában informatikai megoldások üzemeltetése és használata során – ami napjainkban egyre nagyobb figyelmet kap, az IT-biztonság kérdése. A rendszerüzemeltetőknek tisztában kell lenniük a potenciális veszélyekkel, veszélyforrásokkal és fel kell készülniük egy esetleges támadásra, annak kezelésére. Sokszor hallhatunk a rendszeres biztonsági mentések fontosságáról, és ezek típusairól, követelményeiről. Egy jól bevett gyakorlat például az úgynevezett *3-2-1* mentési stratégia, ami egy jó kiindulási alapul szolgálhat minden szervezet számára a biztonsági mentésekhez [16]. A megoldás elnevezése az alábbi elvekből származik:

- három példány az adatokról,
- két különböző eszközön (akár más típusú adathordozókon, pl. SSD, HDD, mágnesszalag – ez segít az adathordozóra jellemző esetleges hibák hatásának csökkentésében),
- egy példányt földrajzilag különböző helyen tároljunk (pl. a cég székhelyén legyenek az eredeti adatok és még egy mentés, további egy példányt pedig tároljunk adatközpontban, vagy vegyünk igénybe harmadik féltől biztonságimentés-szolgáltatást) [10].

A biztonsági mentések elvégzésére és automatizálására többféle megoldást választhat a szervezet az igényeihez igazodva. Bevett szokás például, hogy a cégen belüli mentéseket valamilyen mentést támogató vagy akár teljesen automatizáló szoftverrel oldják meg (például Bareos, Bacula, BackupPC). Az ilyen megoldások üzembe helyezése nehezebb lehet, mintha például csak egyéni scriptekkel hajtanánk végre a mentéseket, de hosszabb távon mégis célszerű lehet megfelelően konfigurálni őket, mert könnyebben kezelhetővé teszik egy komplex infrastruktúrában található gépek adatainak mentését, illetve az egyszer megírt konfigurációs fájlok több gépen is felhasználhatóak. Mindezek mellett ezek a szoftverek általában rendelkeznek valamilyen grafikus felhasználói felülettel (pl. webes interfész), amely tovább egyszerűsíti a mentések készítését, valamint szükség esetén a mentés visszaállítását.

A legtöbb hétköznapi felhasználó számára ismeretlen vagy meglepő lehet, hogy maga az internet és az ezen keresztül elérhető szolgáltatások – gondoljunk például az Ügyfélkapura vagy az internetbank-szolgáltatásokra – nagyon komplex rendszerek nem csak szoftveres, hanem informatikai infrastruktúra szempontjából is. A legtöbb ilyen szolgáltatás egy adatközpontban lévő szerveren fut, ami a beérkező kérésekre ad válaszokat. Ezt a folyamatot úgy is felfoghatjuk, hogy az ilyen szolgáltatások felhasználói lényegében az adott



2.1. ábra. A Bareos biztonsági mentéseket végző program webes felületének részlete. Forrás: `https://www.bareos.com/bareos-webui-modules/`

szolgáltató (a fenti példánál maradva a Magyar Állam és az adott bankok) számítógépeivel kommunikálnak.

Ezek a szervergépek több lényeges különbséggel is bírnak a személyi számítógépekkel szemben. Egyik legfontosabb tulajdonságuk, hogy hibatűrőek bizonyos hardverhibákat illetően: szinte minden főbb komponensből legalább kettő áll rendelkezésre, így ha az egyik meg is hibásodik, akkor a hiba elhárításáig a beépített redundancia miatt a gép képes tovább funkcionálni, általában a felhasználók felé észrevétlenül, míg a gép üzemeltetői figyelmeztetést kapnak a hiba típusáról és a kapcsolódó tennivalókról.

3. fejezet

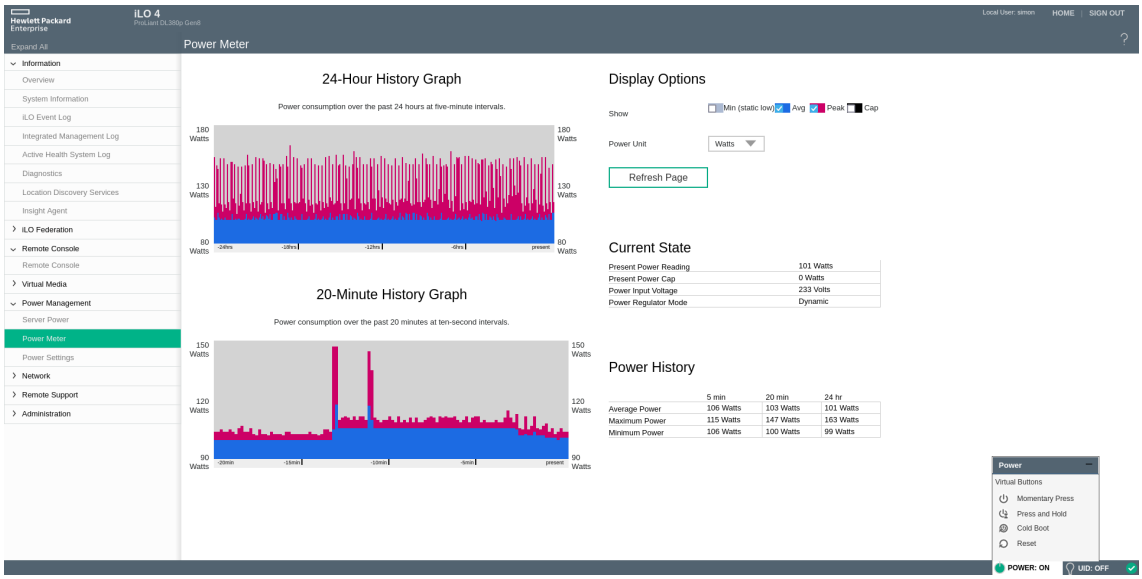
Technológiai áttekintés

3.1. Szervergépek

A szerverek esetében jelentkező, egyénitől nagyban különböző felhasználási körülmények a szerverszámítógépek esetén hardveres szempontból is más felépítést igényelnek. A magas rendelkezésre-állás (high availability, HA) és a modularitás, valamint az ezzel járó könnyű javítások támogatása érdekében az ilyen célra kialakított számítógépek főbb komponensei redundánsak, azaz egy-egy ilyen komponens kiesése nem jelent szolgáltatáskiesést. A meghibásodást a rendszer egyértelműen jelzi magán a gépházon is (általában hibajelző LED-ek segítségével), valamint a menedzsment portjain is. A legtöbb ilyen gép ugyanis rendelkezik egy beágyazott rendszerrel, ami lehetővé teszi a távoli kezelésüket egy webes felületen és SSH-n keresztül még akkor is, ha a szervergép ki van kapcsolva. Ezek lehetőséget biztosítanak a gép legfontosabb mérőszámainak követésére, virtuális kijelző csatlakoztatására, telepítőfájlok felcsatolására, valamint a gép ki- és bekapcsolására.

A fent ismertetett üzemeltetést, karbantartást könnyítő felépítés mellett általában elmondható, hogy az ilyen gépek jelentős része virtualizációra van tervezve – persze ezektől különböző felhasználási módok is jelentkeznek (például fájlszerverek tervezése során a teljesítmény helyett a minél nagyobb tárkapacitásra és adatátvitelre helyezték a hangsúlyt). A dolgozat szempontjából viszont a nagyvállalati környezetben domináló virtualizációs felhasználási terület lesz a lényegesebb, így a továbbiakban az ilyen számítógépekre fogok koncentrálni.

A virtualizációs hosztgépek jellemzője, hogy számos processzorral rendelkeznek, valamint felhasználói szemmel szokatlanul nagy memóriaterülettel bírnak. Ki fog derülni azonban, hogy 12-24 processzormag és akár több száz gigabyte RAM is szűkös erőforrássá válhat egy virtuális gépeket futtató számítógép esetében, hiszen gyakorlatilag itt egyetlen



3.1. ábra. Szervergép fogyasztásának grafikonja egy HPE számítógép távoli menedzsment felületén. Vegyük észre a jobb alsó sarokban megjelenő menüt, amivel lehetőségünk van a gép kikapcsolására és újraindítására is.

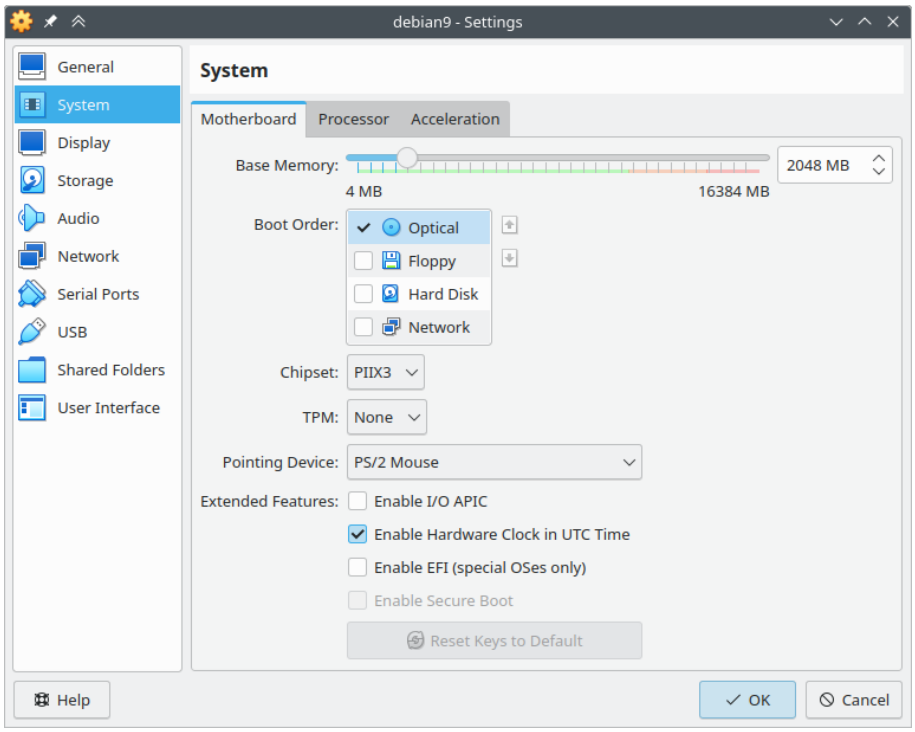
szervernek kell elbírnia akár több tíz számítógép terhelésével is. Ezek mellett általában több (8-24) háttértár-foglalattal is rendelkeznek, melyekhez hardveres RAID-támogatást is adnak.

3.2. Virtualizáció

3.2.1. Áttekintés

A virtualizáció egy olyan technológia, amely lehetővé teszi, hogy egy fizikai számítógépen (az úgynevezett virtuális host-on) több, akár a hosztgép rendszerétől eltérő operációs rendszert futtassunk. Ehhez szükség van egy hypervisor-ra, ami az operációs rendszer legfőbb virtualizációt támogató komponense [12]. Ez a szoftver közvetlenül a szerver hardverén fut, és ez biztosítja a virtuális gépek számára szükséges hardveres erőforrásokat virtualizált hardverinterfészeken keresztül. Szintén a hypervisor felelős az erőforrások kezeléséért, tehát ennek a komponensnek kell biztosítania a virtuális gép számára beállított mennyiségű memória, processzor és háttértár rendelkezésre állását is. Ezek a beállítások a legtöbb modern hypervisor esetében elvégezhetőek grafikus felületen is, erre ad példát a 3.2 ábra.

Napjainkban a legelterjedtebb hypervisorok közé tartozik a VMware ESXi, a XEN és a KVM, melyek részletesebb bemutatását a 3.2.3 alfejezet tartalmazza.



3.2. ábra. Virtuális gép beállításainak részlete a VirtualBox virtualizációs szoftverben.

A virtualizáció számos előnnyel járhat az infrastruktúra és a kiszolgálni kívánt alkalmazások szempontjából. Az egyik legnagyobb ilyen előny például, hogy a virtuális gépek egymástól izoláltan futnak, azaz nincs közvetlen kapcsolat közöttük, ami biztonsági és kezelési, tesztelési szempontból (pl. egy adott csomag vagy szoftver kipróbálásához készíthetünk egy teszt virtuális gépet, amit egyszerűen törölhetünk a teszt végeztével; a telepített program eltávolítása hagyományos környezetben futtatva sokkal körülményesebb lenne) is kedvező lehet. Hasonlóan előnyökkel jár, hogy a legtöbb modern hypervisor lehetőséget biztosít bizonyos erőforrások úgynevezett hot swap-pelésére. Ez azt jelenti, hogy egyes komponenseket (pl. memória) úgy is bővíthetünk, hogy a rendszert nem szükséges ehhez leállítanunk.

3.2.2. Paravirtualizáció és teljes virtualizáció

3.2.3. Virtualizációs lehetőségek összehasonlítása

3.3. Virtualizáció

A fent említett megnövekedett forgalom kiszolgálását hatékonyan lehet kezelni úgy, hogy olyan fizikai számítógépet helyezünk üzembe, mely több, egymástól független operációs

rendszer futtatására is alkalmas. Ilyenkor ezeket a fizikai gépen futó rendszereket virtuális gépeknek (virtual machine, VM) nevezzük. Egy virtuális gép elkülönített erőforrásokat kap a fizikai géptől, hozzáférhet például bizonyos mennyiségű processzormaghoz, memóriához, illetve külön háttértár-partíciói is lehetnek. A virtualizált hardverek és operációs rendszerek a legtöbb esetben a külvilág felé nem különböztethetőek meg a fizikai számítógépektől, és ezzel a megoldással jelentősen csökkenthető a rendszerek és a hozzájuk szükséges informatikai infrastruktúra üzemeltetésének költsége.

A virtualizáció nagy ereje abban rejlik, hogy bizonyos hardverek virtualizációjával egységnyi teljesítményt olcsóbban kaphatunk meg, mintha külön fizikai gépeket helyeznénk üzembe, illetve nagyobb rugalmasságot kapunk a kezelésükben, üzemeltetésükben. Képzeljük el, hogy megveszünk egy számítógépet, amin szeretnénk futtatni egy számunkra fontos alkalmazást, mondjuk a honlapunkat. Ilyenkor az ezen a gépen futó operációs rendszer teljes mértékben megszabhatja, hogy milyen erőforrásokból mennyit használ. Ha egy másik szolgáltatást – például levelezőszervert – szeretnénk emellett futtatni, akkor limitáltabbak a lehetőségeink, hiszen a korábban telepített webszerver már foglal bizonyos erőforrásokat, illetve a program függőségeit és konfigurációs fájljait is telepítettük már, ami esetleg negatívan hat a levelezőszerverünk működésére. Ha mindezt virtualizált környezetben tesszük meg, akkor a topológia megváltozik: a két alkalmazás teljesen elkülönítetten, egymás zavarása nélkül, különböző virtuális gépekben futhatnak, míg magán a fizikai gépen egy úgynevezett hypervisor látja el az erőforrások ütemezésének és kiosztásának (pl. processzoridő, memória) feladatát.

3.3.1. Népszerű virtualizációs technológiák

Mivel a virtualizáció napjainkban nagyon elterjedt technológia, számos olyan megoldás született, mely egyszerűsíti a virtuális gépek üzemeltetését. Ezek közül nagy ismertségnek örvend az Oracle VirtualBox és a VMware Player, azonban ezek a megoldások nem skálázódnak annyira jól, mint a továbbiakban tárgyalt társaik, melyek sokkal megfelelőbbek nagyvállalati szerverkörnyezetben való alkalmazásra. Ezek a megoldások lehetőséget biztosítanak a virtuális gépek távoli elérésre, kezelésére, egyszerűbb telepítésükre, valamint szükség esetén elosztott működésükre.

Ilyen nagyvállalati környezetben is kedvelt megoldás például a VMware ESXi, amely egy igen modern hypervisor számos kényelmi funkcióval ellátva (lehetőség van például a rendszer webes felületről való kezelésére és virtuális gépek sablonból való gyors (nagyságrendileg 5-10 perc) telepítésére). Egy másik kedvelt megoldás az ESXi-vel ellentétben

teljesen ingyenesen, GPLv2-es licenc alatt elérhető XEN hypervisor, mely ugyan kevesebb kényelmi funkciót tartalmaz, de szintén népszerűségnek örvend széleskörű támogatása, kedvező teljesítménye és szabad szoftver voltából eredő ingyenessége miatt. A XEN a 2014 márciusában kiadott 4.4-es verzió óta stabilan működik együtt a libvirt virtualizációs API-val, amely nagyban megkönnyíti a hypervisorral való kommunikációt a virtuális gépek konfigurálása során [15].

A XEN-hez hasonlóan szabad szoftver licenccel érhető el a Kernel-based Virtual Machine (KVM) is, mely a XEN-nél modernebb megoldásnak tekinthető, és manapság széles körben használják a Linux kernelbe való integráltságának és stabilitásának köszönhetően. Bár maga a KVM nem tartalmaz ilyet, de számos interfész elérhető az ezen keresztül futtatott virtuális gépek kezelésére (például virt-manager), valamint akadnak olyan megoldások is, melyek a KVM-re alapozva nyújtanak szélesebb körű virtualizációs megoldást, ilyen lehet[1] például a Proxmox és a Cockpit.

3.3.2. Virtuális gépek használatának néhány előnye

A virtualizáció számos előnnyel bír a szerverinfrastruktúra karbantartása, könnyű kezelhetősége szempontjából, ebből néhány fontosabbat szeretnék kiemelni.

3.3.2.1. Erőforrások testreszabása

Amikor több tíz vagy több száz szerver üzemeltetéséről van szó, akkor hatványozottan számításba kell vennünk az egyes gépekre jutó költségeket. Virtuális gépek esetén ez azért kedvezőbb egy fizikai gépnél, mert ugyan a nagyvállalati környezetbe szánt szervergépek jelentősen drágábbak a személyes felhasználásra tervezett társaiknál, de akár több tíz virtuális gép egyidejű futtatását is lehetővé teszik. Ezáltal az egy fizikai gépre eső, asztali gépeknél megszokott áramfogyasztáshoz képest jóval nagyobb energiafelvétel sokkal kedvezőbb arányt mutat, ha számításba vesszük a futtatott virtuális kiszolgálók számát is.

Mindezek mellett a nagyvállalati felhasználáshoz tervezett számítógépek jóval hibatűrőbbek, hiszen a főbb komponensek redundánsan lettek kialakítva: ezekből az ilyen szerverekben legalább kettő van, és a rendszer automatikusan képes detektálni a hardveres hibákat, és ezek figyelembe vételével tovább működni. További előny lehet még hardveres hibák esetén, hogy ezek a számítógépek széles körben támogatják az úgynevezett *hot swappingot*, amely azt jelenti, hogy bizonyos hardverelemek (általában például háttértá-

[1] A konfigurációtól függően akár többfajta virtualizációs környezet is beállítható, de a KVM az egyik legjobban támogatott.

rak és memóriamodulok) a számítógép bekapcsolt állapotában, annak működése közben is szolgáltatás nélkül cserélhetőek.

Előnyös lehet továbbá, hogy a virtuális gépek erőforrásai szabadon módosíthatók, így akár két újraindítás között is változtathatjuk a rendelkezésre álló memória mennyiségét vagy épp a processzormagok számát. Sőt, egyes hypervisorok és operációs rendszerek ezen erőforrások futásidejű megváltoztatását is támogatják bizonyos korlátozások mellett, így gyakorlatilag a fontosabb virtualizált erőforrások is hot swappelhetőnek tekinthetőek.

3.3.2.2. Snapshotok

Egy másik kedvező lehetőség virtuális gépek használata esetén az, hogy úgynevezett snapshotokat készíthetünk róluk. Ezek a snapshotok a gépet egy adott pillanatbeli állapotban reprezentálják, és később ezeket az állapotokat visszaállíthatjuk, ha szükségünk lesz rá. Egyes megoldások a memóriakép mentését is támogatják, így akár egy futó gép is könnyen visszaállítható. A snapshotok készítése hasznos lehet például rendszerfrissítések esetén, így ha valamiféle hiba lép fel a frissítés során, vagy egy adott szoftver nem megfelelően működik azt követően, akkor a frissítés előtt készített snapshotra visszaállva újra teljes értékűen üzemelhet a szerver, amíg a frissítés során fellépő hibát elhárítjuk.

3.3.2.3. Migráció

Részben az előző ponthoz kapcsolódik a virtuális gépek migrációja. Ez a funkció azt jelenti, hogy egy adott fizikai gépről, mely virtuális gépeket futtat (virtual host), készíthetünk egy snapshotot, amit áthelyezhetünk egy másik virtual hostra, és a virtuális gép ezen futhat tovább egyéb újrakonfigurálás nélkül. Lehetőség van azonban a háttértárak tartalmát elhagyva is átmozgatni egy VM-et egy másik hosztra. Ehhez bevett szokás leírófájlok használata, mely egy virtuális gép konfigurációját tartalmazza. A leírófájlt egy másik hosztgépre áthelyezve ott újra elindíthatjuk a definiált virtuális gépet. Ilyenkor szükség lehet a VM háttértárainak inicializálására, de ettől eltekintve a konfiguráció szabadon hordozható virtual host-ok között. Ilyen migrációra egyes megoldások fejlettebb támogatást is adnak, így akár valós időben, az aktuális terheltség figyelembe vétele mellett automatikusan is áthelyezhetőek virtuális gépek a megadott fizikai hosztok között.

3.3.3. Teljes virtualizáció és paravirtualizáció összehasonlítása

3.3.4. Konténerizáció

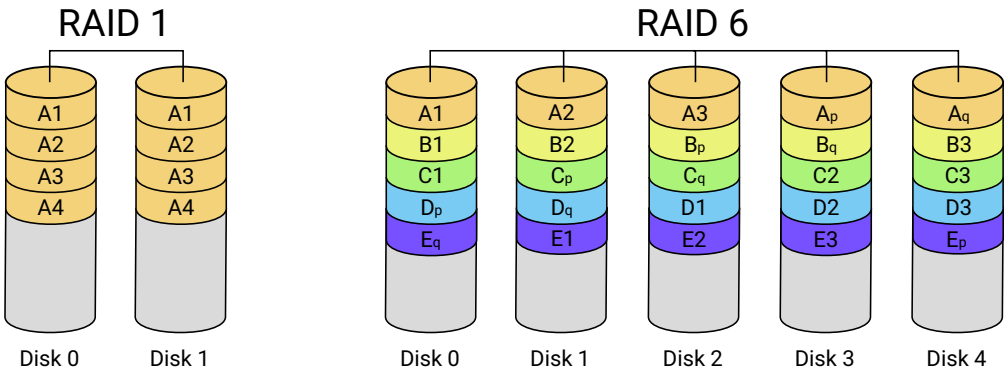
3.4. Logikai kötetkezelés

3.4.1. Snapshotok, mentések készítése

[TODO: vmware a könnyű kezelhetőség és jó támogatás miatt magasan az élen áll; XEN, KVM]

3.5. RAID

Nagyvállalati környezetben nem hagyhatjuk ki a RAID-megoldásokat (a népszerű RAID 1 és RAID 6 megoldásokat a 3.3 ábra szemlélteti), ha adatok biztonsági mentéséről beszélünk. Ezek arra adnak lehetőséget, hogy az adatokat több háttértáron (pl. merevlemez vagy SSD) tároljuk úgy, hogy egy esetleges diszk hiba ne okozzon fennakadást a működésben. Fontos tisztában lenni azonban azzal, hogy a RAID-megoldások nem védenek bizonyos veszélyek ellen (például zsarolóvírusok, fájlok korruptálódása), hiszen az adatok duplikálása valós időben történik, így egy esetleges támadás során a RAID pool-ba bevont összes diszken megváltoznak az adatok, így nem alkalmas a támadás utáni visszaállításra. Emiatt egy RAID pool a *3-2-1* mentési stratégiát alkalmazva csak egyetlen eszköznek tekinthető, hiába több lemezt használunk a mentés során. RAID-elést tehát csak hardveres hibák ellen érdemes használnunk, rosszindulatú támadás esetén ezek nem nyújtanak védelmet az adataink számára.



3.3. ábra. RAID 1 és RAID 6 megoldások felépítése [14]

3.6. Logikai kötetkezelés

Mind a fizikai, mind a virtuális gépek esetén szükség lehet háttértárakra az adatok perzisztens tárolása érdekében. Hagyományos particionálási megoldásokkal hamar nehezen kezelhetővé válhatnak a különböző csatolási pontok és a virtuális gépek számára kiosztott kötetek. Az ilyen problémák elkerülésére jött létre a logikai kötetkezelés, mely a tárhely-virtualizáció egy formája. A logikai kötetkezelésnek több implementációja létezik. Ezek közül jelenleg a Linux kernelben elérhető Logical Volume Manager (LVM)-et fogom részletesen ismertetni.

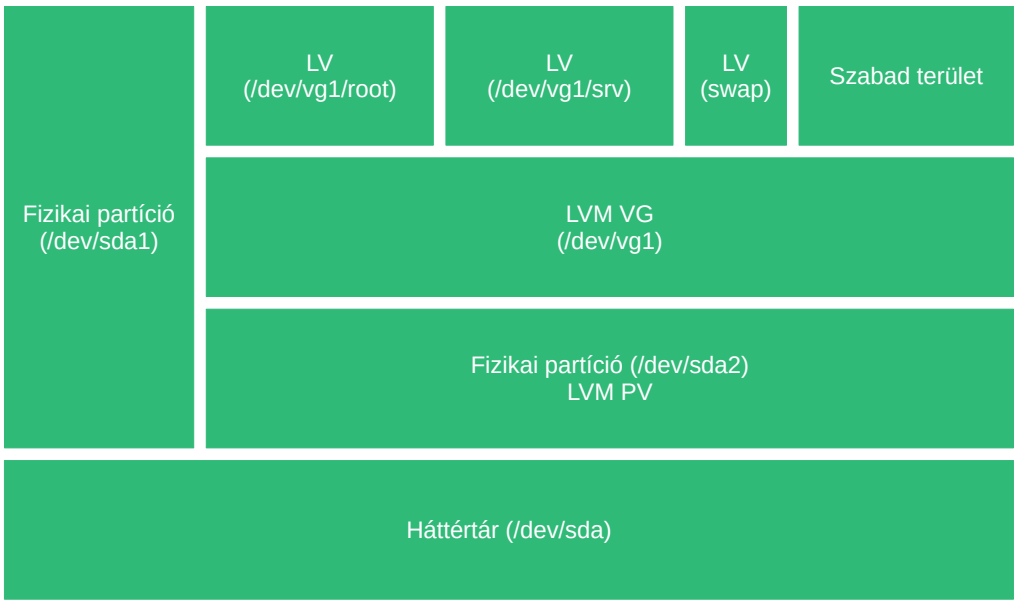
A Linux logikai kötetkezelője három lényegi rétegből áll: a fizikai kötetből (physical volume, PV), a kötetcsoportból (volume group, VG) és a logikai kötetekből (logical volume, LV). Ezt a felépítést a 3.4 ábra szemlélteti egy egyszerű LVM-konfiguráción keresztül. Lehetőség van ennél összetettebb kötetkiosztás létrehozására is, például egy kötetcsoport több fizikai kötetből is állhat, amik akár külön háttértáron is lehetnek, sőt, RAID-csoportot is megadhatunk egy LVM-partíció alapjául. Ezen megoldások használata azonban sok hátránnyal járhat (pl. diszkhiba esetén nehezebb visszaállítani a partíciót), ezért ennek használata alapvetően nem ajánlott [8].

Az LVM tehát úgy épül fel, hogy egy vagy több háttértáron létrehozunk hagyományos fizikai partíciókat, melyek az LVM PV-k alapjául fognak szolgálni. Ezt követően létrehozzuk a kötetcsoportokat az általuk használandó LVM fizikai kötetek megadásával. Az így létrejött csoportban már tudunk létrehozni logikai köteteket, amíg van szabad hely a VG-ben.

Láthatjuk, hogy az LVM-kötetek használata kezdetben több feladattal jár, mint a hagyományos partíciók esetében, azonban hosszabb távon számos előnnyel jár. Talán a logikai kötetkezelés legnagyobb előnye, hogy szabadon allokálhatunk tárterületet a létrehozott köteteknek: ha azt tapasztaljuk, hogy az egyik köteten kevés a szabad hely, akkor fájlrendszertől függően elég lehet akár egy parancs kiadása is ennek kiterjesztéséhez. Lényeges, hogy a hagyományos partíciók használatával ellentétben a logikai kötetkezelés használatakor figyelmen kívül hagyhatjuk a partíciók elhelyezkedésének sorrendjét, így nem szükséges figyelembe vennünk, hogy az adott partíció előtt vagy után van-e szabad tárterület. A megnövelt kötet helyes fizikai háttértárra képzéséről a logikai kötetkezelő fog gondoskodni számunkra. Fontos megjegyezni, hogy a kötetbővítés online is elvégezhető, azaz nem szükséges a kötetet lecsatolni a gépről az átméretezéshez. Ez különösen fontos lehet például a root (/) partíció növelése során, hiszen ezt csak a számítógép leállítása mel-

lett tudjuk biztonságosan lecsatolni. Előállhat olyan helyzet is, hogy egy másik (nem root) partíciót kell online átméreteznünk, például ha azt tapasztaljuk, hogy egy adatbázisszerveren hirtelen nagy mértékben nőtt a tárolt adat mérete. Ilyenkor nincs lehetőség a szerver leállítására, hiszen ez esetben az alkalmazások nem tudnák használni az adatbázist a leállás idejére. Az ehhez hasonló helyzetekre is jó megoldást nyújt a logikai kötetkezelő egy megfelelő, online átméretezést támogató fájlrendszer (pl. XFS, Btrfs) használata mellett. Érdemes megjegyezni, hogy bár az LVM és például a Btrfs-fájlrendszer nyújt támogatást a növelésen kívül a fájlrendszer méretének csökkentésére is, ez a művelet általában nem biztonságos, és adatvesztéshez vezethet. Emiatt érdemes eleinte csak kisebb tárterületet adni a köteteinknek, hiszen kiterjeszteni sokkal egyszerűbb őket, mint csökkenteni a méretüket. Ennek megkönnyítésére is ad lehetőséget az LVM, megadhatjuk, hogy egy kötet egy bizonyos arányú tárhelyhasználat után automatikusan bővüljön, így elkerülve annak betelését.

Az LVM hasznos funkciói közé tartozik még a kötetpillanatképek (volume snapshots) készítésének lehetősége. Ez azt jelenti, hogy a kötetkezelő képes az adott kötet adott pillanatbeli helyzetének rögzítésére, és erre a verzióra szükség szerint visszaállhatunk (rollback). Ez hasznos lehet például nagyobb konfigurációs változások eszközölése esetén, gyorsan változó adatokkal dolgozó rendszerek (pl. adatbázisszerver) biztonsági mentéseinek készítése során, illetve rendszerfrissítések előtt.[2]



3.4. ábra. Egyszerű LVM-kötetkezelési hierarchia.

[2]Egyes eszközök és operációs rendszerek (pl. openSUSE-verziók a snapper-rel (`https://doc.opensuse.org/documentation/leap/reference/html/book-reference/cha-snapper.html`) automatikusan készítenek snapshotot a frissítések telepítése előtt, így hiba esetén visszaállhatunk a frissítés előtti verzióra.

3.7. OS-lehetőségek

Egy nagyvállalati informatikai infrastruktúrában nagy szerepe van a választott operációs rendszernek is, ugyanis nem mindegy, hogy a több száz számítógépből álló rendszerünket mennyire hatékonyan tudjuk karban tartani, egy kritikus biztonsági frissítést milyen hamar tudunk telepíteni az érintett eszközökre, és probléma vagy különleges igény esetén milyen támogatásra számíthatunk a szoftvereinket illetően. Ezeket a szempontokat figyelembe véve manapság elsősorban a Debian, Ubuntu, Red Hat Enterprise Linux és SUSE Linux Enterprise diszribúciók közül választanak a vállalatok.

A Debian stabilitása miatt népszerű választás elsősorban kisebb (néhány tíz gépből álló) infrastruktúrák esetében, viszont a stabilitás az elérhető csomagok verzióinak rovására megy, általában a legújabbnál néhány verzióval régebbi csomagokat szállítanak a disztribúcióval. A Debian előnye, hogy teljesen ingyenesen elérhető, és bár nincs hozzá hivatalos támogatás, harmadik féltől vásárolhatunk ilyen szolgáltatást.

Az Ubuntu egy Debian-alapú operációs rendszer, melyet a Canonical Ltd. fejleszt, és vállalati támogatást is nyújt az OS-hez amellett, hogy az alapverzió ingyenesen érhető el. Előnye, hogy mivel mind szerver, mind pedig asztali környezetben elterjedt rendszer, számtalan projekt és gyártó adja ki a szoftvereit Ubuntu rendszerekre.

A Red Hat és a SUSE Linux-verziók már inkább egy magasabb kategóriát céloznak meg: fő célközönségük a több száz, illetve több ezer gépes környezetet üzemeltető vállalatok, és a fent említett két disztribúciónál alapesetben (a legkisebb támogatási csomagban) is szélesebb körű támogatást biztosítanak az operációs rendszerekhez. Kiemelendő, hogy ez a két disztribúció egyedülálló a biztonság területén: számos biztonsági tesztnek vetették alá őket különböző szervezetek (köztük például kormányzatok és IT-biztonságra specializálódott cégek is), melyeket követően a kereskedelmi forgalomban lévő Linux-disztribúciók közül a legmagasabb minősítéseket és tanúsítványokat kapták meg ezek a rendszerek [7] [11].

Lényeges különbség még, hogy az utóbbi két operációs rendszer RPM-alapú csomagkezelőt használ, mely a Debian és Ubuntu által használt DEB formátumhoz képest több lehetőséget biztosít például javítások (patchek) telepítésére. Emellett ez a formátum általában jobb támogatottságot élvez vállalati szoftverek esetében, ezért ezekben a felhasználási körökben az RPM-csomagokat használnak a DEB-csomagokkal szemben.

3.8. Eszközmenedzsment

3.8.1. Ansible és Salt

3.9. Monitoring

4. fejezet

Virtualizációs környezet létrehozása

4.1. Kialakítani kívánt környezet meghatározása

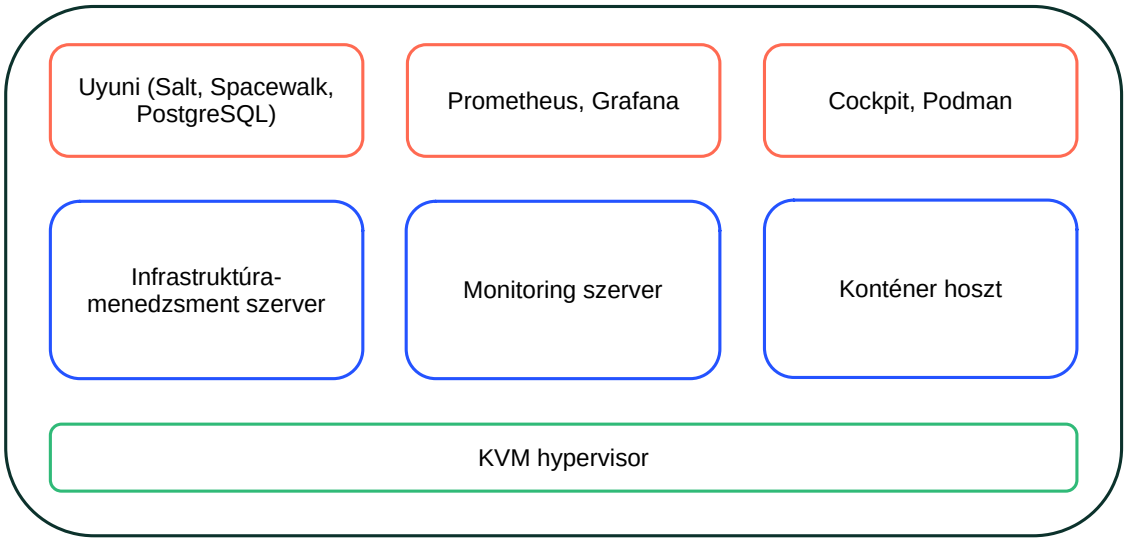
Dolgozatomban egy kisebb léptékű, de a fontosabb elvek ismertetését kellő mértékben lehetővé tevő tesztkörnyezetet fogok kialakítani és részletesen bemutatni. A tesztkörnyezetben egy fizikai gépen (virtual host) fogok virtuális gépeket kialakítani a KVM hypervisor és a libvirt virtualizációs API segítségével. Ezen környezet célja, hogy betekintést engedjen a nagyvállalati környezetekben alkalmazott virtualizációs rendszerek kialakításának fontosabb lépéseibe.

A KVM-re és a libvirt-re azért esett a választásom, mert ezek modern technológiáknak tekinthetőek, az elmúlt 20 évben jöttek létre, és a mai napig aktívan fejlesztik őket. A KVM a Linux kernel része, így a támogatottsága egyedülálló, és lényegében minden Linux disztribúción használható. Emellett több nagy szoftvergyártó és felhőszolgáltató (pl. Google, Red Hat) is a KVM-re építi a saját infrastruktúráját, így a technológia jövője is biztosnak tekinthető [9] [1]. A libvirt a virtuális gépek könnyű kezelhetőségében segít, mivel az API-t több fontos virtualizációt kezelő szoftver (pl. virt-manager, virsh, virt-viewer) is implementálja, így egyaránt biztosított a VM-ek grafikus és a konzolos felületen való kezelése is. Ezek mellett a libvirt számos további kedvező lehetőséget biztosít. Lehetőség van például a virtuális gépek által használt háttértár-partíciók méretének online növelésére, VM-leíró XML-ek generálására, melyek megkönnyítik a virtuális gépek létrehozását, valamint a gépek másik hosztgépre történő áthelyezésében is könnyebbséget jelentenek.

Ezen technológiák lehetőségeit figyelembe véve a tesztkörnyezettel szemben az alábbi elvárásokat támasztottam:

- legyen alkalmas nagyvállalati igények kielégítésére, egy olyan infrastruktúra jöjjön létre, ami nagyvállalati környezetben is megállná a helyét,
- mutassa be a virtualizációhoz és a virtuális rendszerek üzemeltetéséhez kapcsolódó jó gyakorlatokat (pl. particionálás, LVM kötetkiosztás),
- legyen képes a virtualizált OS-környezetben futó programok mellett konténerizált alkalmazások futtatására is,
- legyen központilag kezelhető infrastruktúramenedzsment szoftver segítségével, nyújtson lehetőséget konfigurációs fájlok egységes telepítésére,
- a rendszer működése, teljesítménye legyen jól nyomon követhető monitoring rendszeren keresztül.

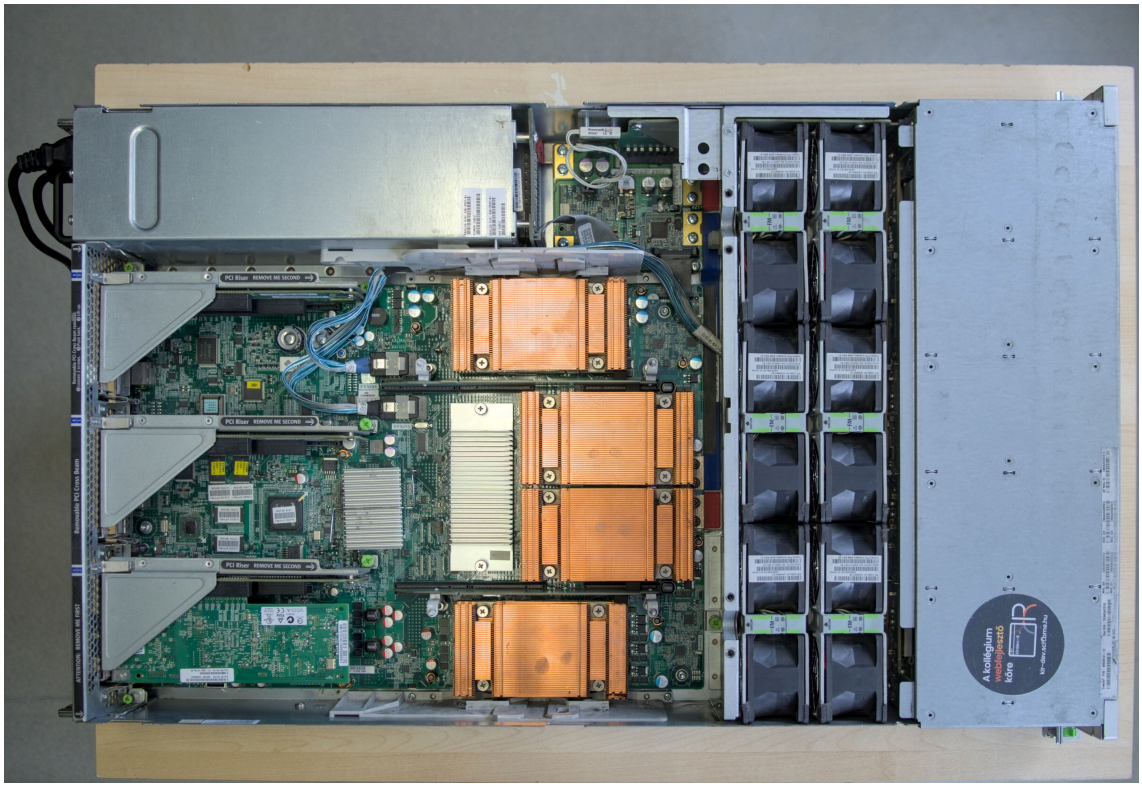
Az így meghatározott tesztkörnyezet felépítését a 4.1 ábra mutatja be. Az ábrán jól elkülöníthetően jelennek meg az architektúra egyes rétegei: a legalsó szinten helyezkedik el a hypervisor, melyre a kékkel jelölt virtuális gépek épülnek, továbbá narancssárgával láthatóak az alkalmazásréteg elemei, melyek az alattuk elhelyezkedő virtuális gépeken futnak.



4.1. ábra. A tesztkörnyezet tervezett felépítése.

4.2. Fizikai gép ismertetése

Ahogy arról a 3.1 alfejezetben már írtam, a szervergépek több lényeges tulajdonságukban is eltérnek a személyi számítógépektől. A virtualizáció szempontjából legfontosabb ilyen különbségek a processzormagok száma és a memória mennyisége. A tesztkörnyezetet szerettem volna egy ilyen gépen megvalósítani, hogy az ténylegesen a lehető legközelebb állhasson egy valós felhasználási környezethez. Bár a lehetőségeim korlátozottak voltak, sikerült beüzemelnem egy régi, Sun Fire X4450 típusú szervergépet. Ez négy fizikai CPU-val rendelkezik, mind a négy processzor 6-6 magot tartalmaz, így összesen 24 maggal gazdálkodhattam. Emellett a gép 64 GB memóriával van felszerelve, és egy 1 TB-os SSD-meghajtó található benne. Ezek mellett a korábban említett hardveres redundancia is megjelenik a gépben: két tápegysége és négy hálózati csatlakozója van, továbbá távolimenedzsment-porttal is rendelkezik, mely lehetővé teszi a szerver távolról történő ki- és bekapcsolását, illetve a rendszernaplók böngészését.



4.2. ábra. A tesztkörnyezetben használt fizikai gép. A fotón megfigyelhető a moduláris felépítés, a memóriatálcát eltávolítva pedig a négy különálló CPU is láthatóvá válik.

4.3. Operációs rendszer

Értekezésemben nagy szerepe lesz a választott operációs rendszereknek, hiszen ezek fognak a virtualizációs rendszer alapjául szolgálni, valamint képesnek kell lennünk a gépek távoli menedzsmentjére is, így mindenképpen olyan megoldásra van szükség, amely jól támogatott a választott infrastruktúramenedzsment eszköz által. Fontos szempont volt továbbá, hogy a tesztkörnyezet a lehetőségekhez mérten jól reprezentálja a nagyvállalati környezetben használatos rendszereket, így sok olyan OS-verzió kikerült a lehetőségek közül, amelyek ugyan népszerűek például asztali megoldásként, de egyes nagyvállalati szoftverek (legyen az adatbázismotor, vagy bizonyos eszközvezérlők, driverek) hivatalosan nem támogatottak rajtuk. Emiatt az operációs rendszerek kiválasztása során körültekintően jártam el, több Linux-disztribúció is szóba került, az ezekről született konklúziót itt foglalom össze néhány mondatban.

4.3.1. OS-kiválasztás folyamata

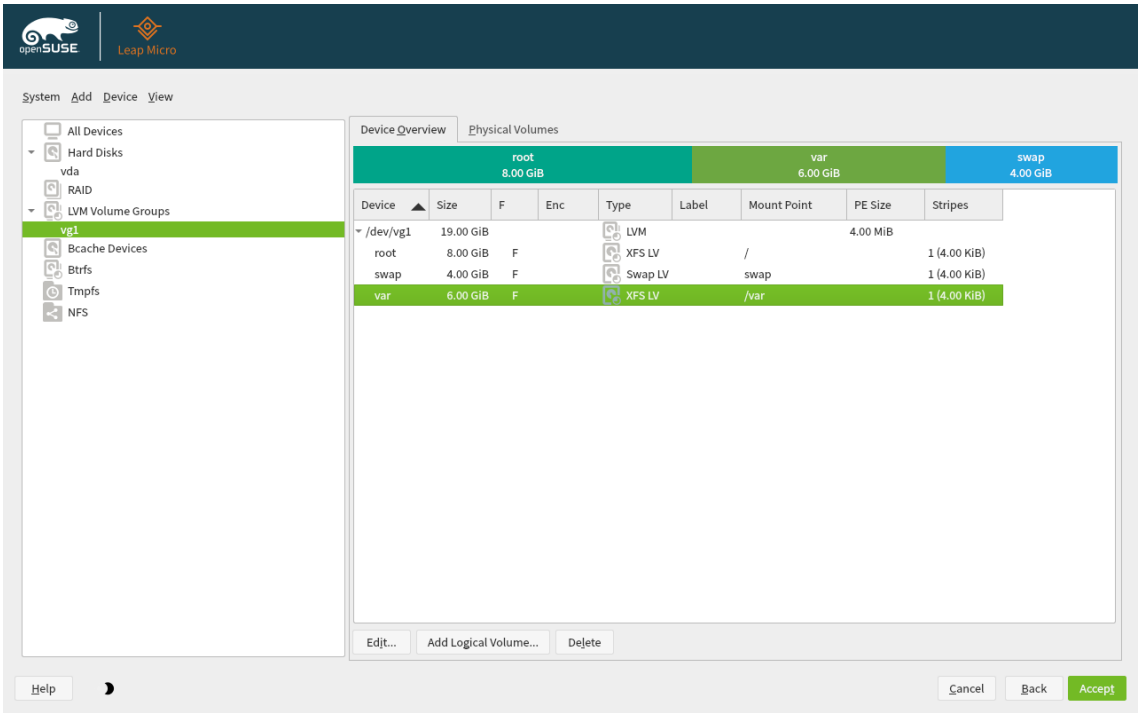
Ahogy a 3.7. alfejezetben is kitértem rá, a nagyvállalatok elsősorban a Red Hat és a SUSE Linux-disztribúciók közül választanak, hiszen ezeknek a velük együtt járó támogatás és a szoftvercsomagok széleskörű támogatottsága miatt kényelmesebb és hatékonyabb az üzemeltetésük, valamint biztonsági szempontból is kedvezőbbek (például gyorsabban kapnak meg bizonyos frissítéseket, patcheket). Szintén jobban támogatottak ezeken a rendszereken a különböző felhasználásspecifikus modulok, például high availability (HA), live patching (támogatás pl. kritikus kernel biztonsági javítások telepítése a számítógép újraindítása nélkül) és real time computing (valós idejű, nagy időbeli pontosságot igénylő alkalmazások futtatására alkalmas környezet).

A fent ismertetett szélesebb körű támogatottság miatt a tesztkörnyezethez használni kívánt operációs rendszerek köre a Red Hat-re és a SUSE Linuxra korlátozódott. A végső döntésben végül az alábbi szempontok segítettek:

- a tesztkörnyezetet szerettem volna egy ökoszisztémán belül tudni mind a virtuális gépeket futtató, mind pedig az azokon futó OS-ek esetében,
- könnyebb konfigurálhatóság: mivel több gépet kellett telepíteni, így fontos szerepe volt annak, hogy egy-egy operációs rendszer telepítése milyen bonyolultságú,

- a környezetet a költségek minimalizálása mellett szerettem volna létrehozni, így lényeges szempont volt, hogy az adott rendszerhez ne kelljen előfizetést vásárolni, mégis a lehető legközelebb álljon a kereskedelmi forgalomban kapható termékekhez.

Mindezek figyelembevételével és korábbi tapasztalataim alapján a SUSE termékcsaládja mellett döntöttem. A támogatással rendelkező, előfizetéses modellt használó nagyvállalati változat mellett szabadon beszerezhető openSUSE operációsrendszer-család megfelelt a tesztkörnyezettel szemben támasztott elvárásaimnak. A rendszer telepítését és a későbbi konfigurációt a YaST keretrendszer segíti, mely számos moduljával (pl. particionálás, hálózati és tűzfalbeállítások) nagyban hozzájárul a gépek könnyebb beállításához, kezeléséhez. A YaST – mivel szervereken való használatra tervezték, melyek gyakran nem rendelkeznek grafikus felülettel – a 4.3 ábrán látható megjelenés mellet egy konzolos, GUI-szerű (GUI-like) felülettel is rendelkezik, így a konfiguráció kényelmesen elvégezhető konzolos hozzáférés, például SSH használata esetén is.



4.3. ábra. LVM-kötetek létrehozása openSUSE Leap Micro telepítése során grafikus YaST telepítő segítségével.

Az openSUSE-projekt több operációs rendszert is fejleszt[1], ezek közül én a tesztkörnyezetben kettőt használtam, melyeket a következő alfejezetekben ismertetek.

[1] `https://get.opensuse.org/`

4.3.1.1. openSUSE Leap

A Leap egy hagyományos értelemben vett szerver operációs rendszer. Gyakran kap biztonsági frissítéseket, új verziói pedig körülbelül évente jelennek meg. Alapjául a SUSE Linux Enterprise szolgál, melynek előnye, hogy a két rendszer csomagjai binárisan kompatibilisek egymással, azaz egy SLE-rendszerre készített csomag garantáltan használható openSUSE Leap-en is, és fordítva [2] [3]. Utóbbi előnye, hogy így számos, a közösség (akár a hivatalos openSUSE projekt, akár a felhasználók) által készített csomagot használhatunk a SLE-alapú rendszerünkön is, bár ehhez nem kapunk hivatalos támogatást.

A nagyvállalati rendszerből való leszármazás másik nagy előnye, ami fontos volt számomra a kiválasztási folyamat során, hogy így gyakorlatilag a SUSE Linux Enterprise egy ingyenes verzióját használhatom, mely lényegében teljesen megegyezik a vállalati környezetben használt megoldással, és előfizetés nélkül is kap frissítéseket, így folyamatosan naprakészen tartható. A biztonsági javításokat illetően fontos megjegyezni, hogy a Leap rendelkezik egy olyan csomagforrással (repository) is, mely a SUSE Linux Enterprise-ban is elérhető frissítéseket tartalmazza, így az ott hozzáférhető fontos javításokat is telepíthetjük a Leap-et futtató rendszereinkre [4].

4.3.1.2. openSUSE MicroOS

A MicroOS egy újfajta megközelítést használó, modern operációs rendszer, mely elsősorban konténerizált alkalmazások futtatásához készült. Az OS előnye, hogy az alap installáció csak egy minimális szoftvercsomagot tartalmaz, így az erőforrásigénye elenyésző. A MicroOS egy írásvédett (read-only) BTRFS gyökérkönyvtárral rendelkezik, melynek előnye, hogy magas szintű támogatást nyújt fájlrendszer-pillanatképek (filesystem snapshots) kezelésére. Erre a technológiára épít a MicroOS filozófiája: atomi frissítéseket támogat, ami azt jelenti, hogy egy csomag vagy frissítés telepítése során nem az éppen használatban lévő partíció változik, hanem egy új snapshotba kerülnek a módosítások, mely – amennyiben a módosítás sikeresen lezajlott – a következő bootolási folyamat során aktívvá válik, és az OS erről kerül betöltésre, így ekkor már használhatjuk a telepített csomagokat. Az atomi frissítések lényege, hogy a módosítások csak akkor lépjenek életbe, ha a teljes folyamat hiba nélkül futott le, azaz például ha egy művelet során a módosítandó 100 csomagból akár csak egy nem tud települni valamilyen hibából eredendően, akkor a teljes telepítés meghiúsul, ezzel elkerülve azt, hogy a rendszer inkonzisztens állapotba kerüljön. A MicroOS ezáltal képes biztosítani azt, hogy a rendszerünk mindig használható állapotban legyen.

A snapshotok fontos tulajdonsága, hogy mindaddig, amíg nem kerülnek törlésre, használatukkal a rendszer bitről bitre visszaállítható abba az állapotba, amiben a pillanatkép készítésekor volt. Ennek nagy jelentősége lehet egy félresikerült rendszerfrissítést követően, hiszen a korábbi állapotra visszaállva a rendszer zavartalanul folytathatja a működést a hiba elhárításáig. A probléma okának felderítését segíti a snapshotok felcsatolásának lehetősége: ez azt jelenti, hogy a BTRFS fájlrendszer képes arra, hogy a éppen használt partíció mellett az ahhoz tartozó pillanatképeket is felcsatoljuk, sőt, a két állapotot össze is vethetjük a verziókezelő rendszerekben megszokott módon (erre például a YaST beépített támogatással rendelkezik), mely tovább könnyítheti a hiba forrásának felderítését.

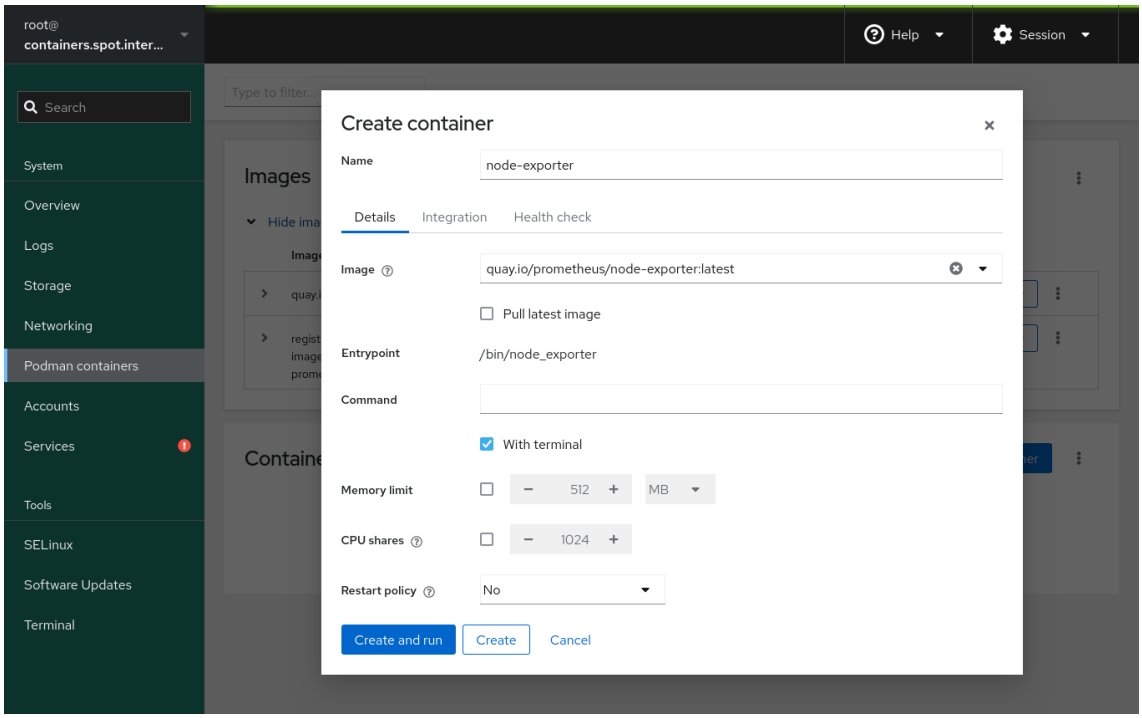
A MicroOS különlegességei közé tartozik még, hogy a szerver operációs rendszereknél megszokott konzolos és távoli asztalos elérés mellett egy webes felületet is biztosít a rendszer kezelésére. Ehhez a Cockpit adminisztrációs rendszert használja, mely az utóbbi években egyre nagyobb népszerűségnek örvendő megoldás. A Red Hat disztribúciói például már ezt a rendszert ajánlják a virtualizáció kezelésére a korábban megszokott virt-manager helyett [5].

A Cockpit felülete gyors áttekintést nyújt a rendszer állapotáról, továbbá könnyíti a konténerek létrehozását (4.4 ábra) és kezelését. A fontosabb metrikák (processzor-, memória-, háttértár és hálózathasználat) megtekintése mellett szükség esetén közvetlenül is be tudunk avatkozni a rendszer működésébe, ugyanis a felület egy terminállal is rendelkezik. Továbbá a futó szolgáltatások állapotát is figyelemmel kísérhetjük, valamint a felhasználói fiókokat is kezelhetjük a Cockpit segítségével.

Az openSUSE-projekt kétféle MicroOS-verziót tart karban: a MicroOS-t, mely egy rolling release modellt követ, azaz a rendszer folyamatosan (akár napi szinten) kapja meg a frissítéseket, így több, kisebb verzióugrással tartható karban, míg az openSUSE Leap Micro a SUSE Linux Enterprise Micro kiadási modelljét követi, és a Leap-hez hasonlóan bináris kompatibilitást garantál a két verzió között. A tesztkörnyezethez a stabilitás és kompatibilitás miatt a Leap Micro változatot választottam.

4.4. Hálózati topológia

Az infrastruktúra működésében fontos szerepe van a hálózatnak: a távoli elérésen túl biztosítani kell a szoftvercsomagok elérhetőségét is, valamint a későbbiekben látni fogjuk, hogy a monitoring rendszer is hálózation keresztül gyűjti az adatokat. Ezek miatt lényeges



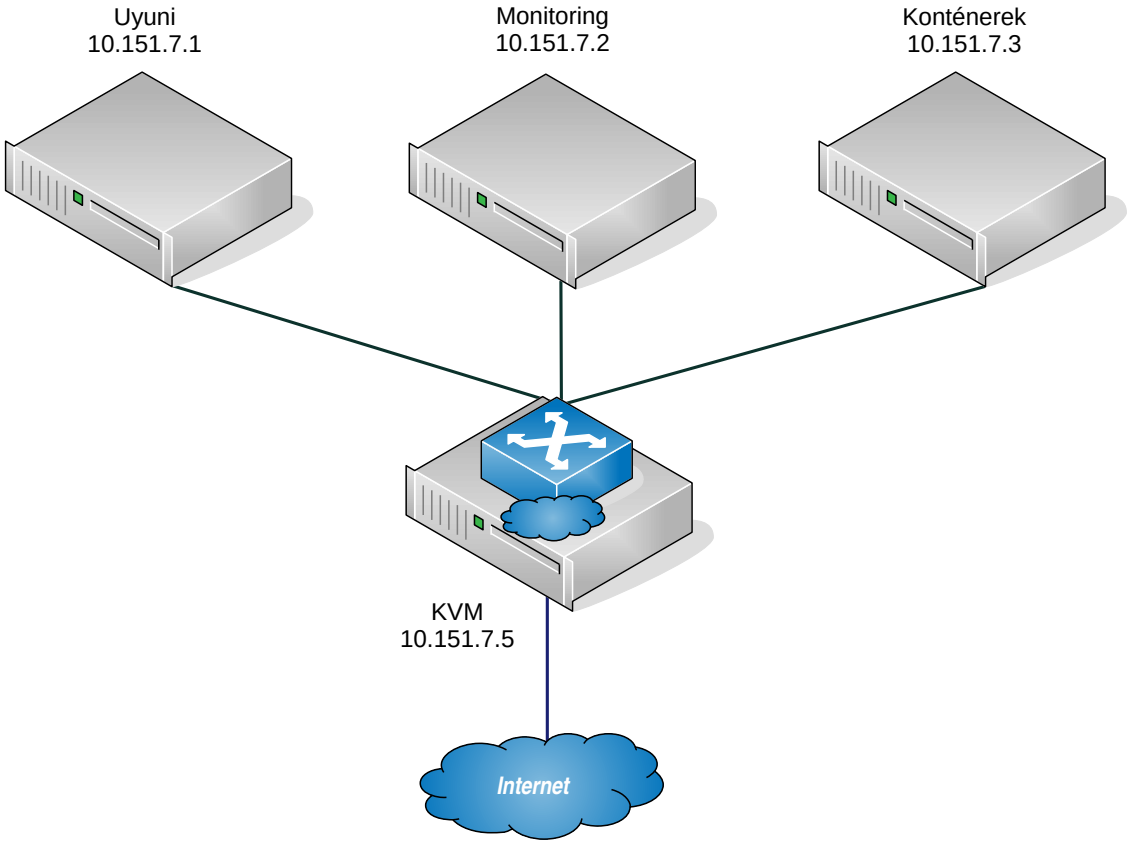
4.4. ábra. Konténer létrehozása openSUSE Leap Micro-n, a Cockpit webes felületén keresztül.

volt, hogy a gépek tudjanak kommunikálni egymással és a külvilággal. A tesztkörnyezet hálózati felépítését a 4.5 ábra szemlélteti.

4.4.1. Bridge-dzselt hálózati interfész

A virtualizált környezetek sajátossága, hogy a virtuális gépek alapesetben egy – a virtualizációt biztosító szoftver által kezelt – hálózatra tudnak csatlakozni, a fizikai gép hálózatán nincs lehetőségük kommunikálni. Ez a megoldás általában külön konfiguráció nélkül elérhető, viszont hátránya, hogy a külvilág felé gyakorlatilag láthatatlanná válik a virtuális gép. Bár ez porttovábbítással és a tűzfalbeállítások, valamint az érintett szolgáltatások módosításával orvosolható, a hálózati interfészek bridge-dzselése egy szélesebb körben használható megoldást nyújt.

Egy bridge-dzselt interfész lehetővé teszi a virtuális gépek számára, hogy a gazdagéppel azonos hálózaton kommunikáljanak, azaz ugyanúgy működjenek, mintha minden virtuális gép virtualizált hálózati interfészéhez tartozna egy dedikált hálózati csatlakozó a fizikai gépen, mely egyazon hálózathoz csatlakozik. Ehhez a gazdagép hálózati beállításaiban létre kell hozni egy hálózati híd eszközt, és a használni kívánt interfészt be kell állítani bridge masterként. Ezeket a beállításokat egyszerűen elvégezhetjük parancssori felületen, illetve a YaST segítségével is (4.6 ábra). Az így létrejött virtuális eszköz az OSI-modell



4.5. ábra. A tesztkörnyezet hálózati felépítése.

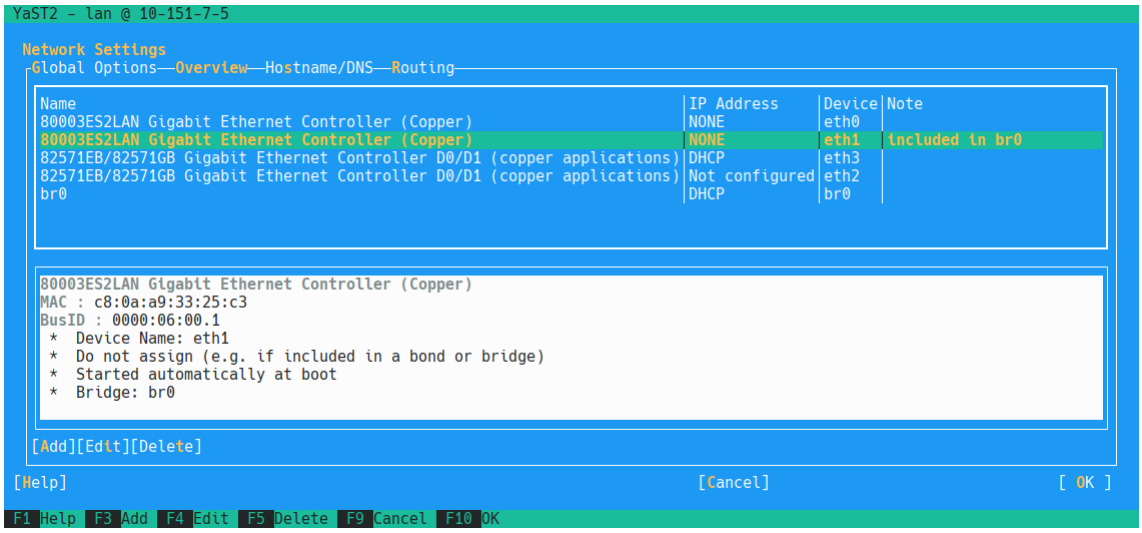
szerinti második szinten, az adatkapcsolati rétegben működik a hardveres switch-ekhez hasonlóan [13].

4.5. Virtualizációs komponensek telepítése

A tesztkörnyezet szoftveres alapját a virtualizációs megoldások adják. Ebben az alfejezetben ismertetem a fizikai gép előkészítését és a virtuális gépek telepítésének folyamatát, valamint az ezekhez kapcsolódó beállítási lépéseket, kitérve például a particionálás folyamatára és a virtuális gépek terminálos elérésére.

4.5.1. Hosztgép konfigurálása

A virtualizációs környezetet futtató számítógépre az openSUSE Leap 15.5-ös verzióját telepítettem, mely a tesztkörnyezet kialakításakor a disztribúció legfrissebb stabil elérhető változata. A telepítés során a gép szerepének (system role) a szerver opciót választottam. Ez a felhasználási céloknak teljesen megfelelt, hiszen ez a megoldás is egy jól felszerelt operációs rendszert telepít, csak asztali környezet nélkül. Mivel a szervert elsősorban kon-



4.6. ábra. A tesztkörnyezethez beállított hálózati bridge alapjául szolgáló eth1 interfész beállításainak részletei a YaST konfigurációs program parancssori változatában. Látható, hogy az interfész a br0 bridge eszközhöz van társítva, és emiatt nincs hozzárendelve IP-cím.

zolos felületen, SSH-n keresztül szerettem volna használni, ezért ez nem jelentett gondot. Sőt, a tesztkörnyezet szempontjából előnnyel is járt: a rendszerre nem volt szükséges az asztali környezet működéséhez elengedhetetlen csomagok telepítése, ami nem csak a tárhellyel való takarékoskodásban segített, de a későbbiekben is könnyítette a karbantartási folyamatokat, mert kevesebb csomagot kellett frissíteni és így az esetleges támadási felület (sérülékenységek száma) is kisebb volt. Lényeges azonban megjegyezni, hogy az X11 könyvtár a szerver csomag részeként is telepítésre került, így adott volt a lehetőség X forwarding[2] használatára.

A hálózatkezeléshez – a virtuális gépek hálózati elérését lehetővé teendő – a 4.4.1. alfejezetben bemutatott bridge-dzselt hálózati interfészt állítottam be. Ehhez az alapértelmezett beállításokhoz képest annyit kellett módosítani, hogy a bridge-dzselt eszköz a fizikai interfészen keresztül kommunikáljon, illetve hogy a virtualizációs hosztnak kiosztott `10.151.7.5`-ös IP-címet ne az `eth1` fizikai interfész kapja meg, hanem az újonnan létrehozott `br0` eszköz. Ezt követően a számítógép a korábban megszokott módon tudott a hálózaton kommunikálni, viszont lehetővé vált, hogy a virtuális gépek is hozzáférjenek a fizikai gép hálózatához a bridge eszközön keresztül.

A hálózaton kívül a másik lényeges tervezői döntés a logikai kötetek (LVM) alkalmazása volt. Ez a gyakorlatban azt jelentette, hogy a boot partíció kivételével minden egyéb kötetet LVM-kötetként hoztam létre. Ennek legfőbb előnye számomra a kötetek méretének

[2]Távoli szerveren futtatott, grafikus felülettel rendelkező alkalmazások ablakának a kliensgép képernyőjén való megjelenítését lehetővé tevő technológia.

rugalmas kezelése volt, melyről a 3.6. alfejezetben írtam bővebben. Emellett lehetőséget biztosít pillanatképek készítésére is, melyek készítése például OS-frissítések előtt lehet releváns, és nagyban megkönnyíti rendszer korábbi állapotának helyreállítást, ha valami hiba jelentkezik a folyamat során. A kötetkiosztás úgy történt, hogy minden virtuális gép kapott egy külön LVM-kötetet, amit egy egyedülálló tárolóeszközként érzékelt, és ezt használhatta az adatok tárolására, akár további particionálás mellett is.

A kezdetleges konfigurációt követően telepítettem a KVM hypervisort és a virtuális gépek kezeléséhez szükséges csomagokat. Ehhez openSUSE-disztribúciókon külön ún. pattern áll a rendelkezésünkre. Ez azt jelenti, hogy nem kell megadnunk minden telepítendő csomagot, hanem elég a `kvm_server` és a `kvm_tools` pattern-ök telepítése, és ezek automatikusan telepítésre jelölik a teljes értékű KVM-szerverhez szükséges csomagokat, emellett néhány hasznos segédprogramot (pl. virt-manager, virsh) is magukkal hoznak. Ezen csomagok telepítését követően minden előfeltétel adottá vált a virtuális gépek telepítéséhez.

4.5.2. Virtuális gépek telepítése

VM-ek telepítéséhez elsősorban a `virt-install` parancsot használtam. Ez a program lehetővé teszi az összes lényeges paraméter megadását, majd távoliasztal-protokoll használatával (alapértelmezetten SPICE, de választhatjuk például a VNC-t is) megjeleníti a virtuális gép kijelzőjét egy ablakban, melynek segítségével személyre szabhatjuk a telepítést és telepíthetjük az operációs rendszert. A távoli asztalon keresztüli elérés csak a megfelelő környezet, pl. SSH használata esetén, X forwardinggal működik. Amennyiben a virt-install nem talál kijelzőt, akkor a telepítés parancssoron keresztül történik. A virt-install sikeres futás esetén egy virtuálisgép-leíró XML-fájlt hoz létre, mely a gép összes paraméterét tárolja (4.3. kódrészlet). A későbbiekben a VM konfigurációjának módosítása esetén ezt a fájlt kell módosítanunk (akár szövegszerkesztővel, akár GUI-n, például virt-manager-rel). A leírófájl a virtuális gép migrációjához is használható.

Azonban még mielőtt a konkrét telepítést elkezdhetnénk, létre kell hoznunk azt a partíciót, melyre a virtuális gép adatai kerülnek. Esetemben ez azt jelentette, hogy az egyes virtuális gépekhez új LVM-köteteket kellett létrehoznom. Mivel a logikai kötetek számára otthont adó partíciót és a kapcsolódó fizikai kötetet és kötetcsoportot már a hoszt OS telepítésekor létrehoztam, ezért a VM-ek telepítése során elég volt csak egy-egy logikai kötetet (LV) létrehoznom, melyhez az `lvcreate` parancsot használtam (4.1 kódrészlet). Emellett telepítési forrást is meg kellett adni, melyhez én telepítő lemezképeket használ-

tam. Ilyenkor a VM telepítésekor fel kell venni egy virtuális CD-meghajtót a géphez, és meg kell adni a telepítési forrás elérési útját.

```
lvcreate -L 20G --name kvm-monitoring vg1
```

4.1. lista. Virtuális gépek logikai kötetének létrehozásához használt parancs.

Ezen túl meg kell adni a gép fontosabb paramétereit is (pl. processzorok száma, memória mennyisége), hogy milyen erőforrásokkal szeretnénk telepíteni azt. A tesztkörnyezetben használt gépek telepítése során elsősorban a dokumentációban ismertetett rendszerkövetelményeket vettem figyelembe az erőforrások meghatározásánál, de mivel aránylag sok erőforrás állt rendelkezésemre, így előfordult, hogy a számítási műveletek gyorsítása érdekében a szükségesnél több magot adtam a virtuális gépeknek. Mivel a VM-ek konfigurációja szabadon változtatható a későbbiekben is (esetleg a gép újraindítása szükséges az érvényre jutásukhoz), ezért ez nem jelentett problémát a későbbiekben sem. Emellett a KVM támogatja a memória és processzor erőforrások overcommit-olását is, azaz nem jelent problémát, ha esetleg a fizikai gépen elérhetőnél több CPU-erőforrást osztottunk ki a virtuális gépek számára, bár ennek használatára a dolgozathoz készített tesztkörnyezetben nem volt szükség [6]. A tesztkörnyezet egyik virtuális gépének telepítéséhez használt parancsot a 4.2 kódrészlet mutatja be.

```
virt-install --name uyuni --memory 32768 --vcpus 12 --cdrom /mnt/openSUSE-Leap-15.5-NET-x86_64-
   Media.iso --os-variant opensuse15.5 --disk /dev/vg1/kvm-uyuni
```

4.2. lista. Virtuális gép telepítése a virt-install segédprogrammal.

```
<memory unit='KiB'>33554432</memory>
<currentMemory unit='KiB'>33554432</currentMemory>
<vcpu placement='static'>12</vcpu>
<resource>
  <partition>/machine</partition>
</resource>
<os>
  <type arch='x86_64' machine='pc-q35-7.1'>hvm</type>
</os>
...
<devices>
  <emulator>/usr/bin/qemu-system-x86_64</emulator>
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='native' discard='unmap'/>
    <source dev='/dev/vg1/kvm-uyuni' index='1'/>
    <backingStore/>
    <target dev='vda' bus='virtio'/>
    <boot order='2'/>
    <alias name='virtio-disk0'/>
```

```
    <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
  </disk>
...
</devices>
```

4.3. lista. Virtuális gép leírófájljának részlete.

A virtuális gép operációs rendszerének telepítése során ki kellett alakítani a kívánt partíciókiosztást, mely a felhasználási körtől függően változott, de alapvetően mindenhol LVM-alapú kötetkiosztást alkalmaztam. Emellett szükséges volt bizonyos hálózati beállítások (pl. DNS-szerver címe, gépnév) módosítása is. Ezeken felül és az alapvető adatok – mint például felhasználói fiókok létrehozása, lokalizációs beállítások konfigurálása – megadásán túl mást nem volt szükséges átállítani a telepítés során. A sikeres installációt követően foghattam hozzá a virtuális gépeken futó szolgáltatások telepítéséhez, melyeket a következő fejezetekben fogok részletesen ismertetni.

4.6. Gépmenedzsment: Salt

4.7. Monitoring

4.8. Továbbfejlesztési lehetőségek

Köszönetnyilvánítás

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Szójegyzék

hot swap Számítógép-komponensek eltávolítása vagy hozzáadása egy futó rendszerhez (annak leállítása vagy újraindítása nélkül). 9

hypervisor Az a szoftver, amely koordinálja a virtuális gépek és az azokat futtató fizikai gép hardvere közötti interakciót. 8, 9, 18, 19, 28, 32

libvirt Nyílt forráskódú virtualizációs API, mely lehetőséget biztosít több hypervisor (pl. KVM, XEN, VMware ESXi) egységes interfészen keresztüli kezelésére. 18

overcommit Hatékonyabb erőforráskihasználást lehetővé tevő technológia egyes hypervisorokban. Használatával a virtuális gépek számára több virtuális erőforrást oszthatunk ki, mint amennyi ténylegesen rendelkezésre áll a gazdagépen. Alapvetően előnyös lehet a használata, mivel a VM-ek általában nem használják ki a nekik biztosított erőforrások 100%-át, de a gazdagép túlterhelése esetén jelentős teljesítményvisszaeséssel járhat, ezért fontos, hogy használata esetén még körültekintőbben monitorozzuk rendszereinket [6]. 29

Betűszavak

API application programming interface. 11, 18, 32

CPU central processing unit. 20, 29

DNS domain name system. 30

HA high availability. 7, 21

KVM Kernel-based Virtual Machine. 11, 18, 28, 29, 32

LV logical volume. 28

LVM Logical Volume Manager. 14, 15, 19, 22, 27, 28, 30

OS operating system. 19, 21, 23, 28

PV physical volume. 14

RPM RPM Package Manager, eredetileg Red Hat Package Manager. 16

SLE SUSE Linux Enterprise. 23, 24

SPICE Simple Protocol for Independent Computing Environments. 28

SSD solid state drive. 20

SSH secure shell. 22, 27, 28

VG volume group. 14

VM virtual machine. 10, 18, 28, 29, 32

VNC virtual network computing. 28

Irodalomjegyzék

[1] Andy Honig, Nelly Porter, Google Cloud: 7 ways we harden our KVM hypervisor at Google Cloud: security in plaintext (2024. május 5.). https://cloud.google.com/blog/products/gcp/7-ways-we-harden-our-kvm-hypervisor-at-google-cloud-security-in-plaintext.

[2] Douglas DeMaio, openSUSE: openSUSE Leap 15.3 Bridges Path to Enterprise (2024. május 4.). https://news.opensuse.org/2021/06/02/opensuse-leap-bridges-path-to-enterprise/.

[3] Jeff Reser, SUSE: Introducing SUSE Linux Enterprise 15 SP3 (2024. május 4.). https://www.suse.com/c/introducing-suse-linux-enterprise-15-sp3/.

[4] openSUSE: openSUSE Leap 15.3 Release Notes (2024. május 5.). https://doc.opensuse.org/release-notes/x86_64/openSUSE/Leap/15.3/#installation-new-update-repos.

[5] Red Hat: Deprecated functionality (2024. május 5.). https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/9.4_release_notes/deprecated-functionality#deprecated-functionality-virtualization.

[6] Red Hat: Overcommitting with KVM (2024. május 8.). https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/chap-overcommitting_with_kvm.

[7] Red Hat: Red Hat Adds Common Criteria Certification for Red Hat Enterprise Linux 8 (2024. április 24.). https://www.redhat.com/en/about/press-releases/red-hat-adds-common-criteria-certification-red-hat-enterprise-linux-8.

[8] Red Hat: Red Hat Enterprise Linux: Logical Volume Manager Administration (2024. április 22.). https://access.redhat.com/documentation/en-us/red_

hat_enterprise_linux/7/html/logical_volume_manager_administration/lvm_components.

[9] Red Hat: Red Hat Virtualization (2024. május 5.). https://access.redhat.com/products/red-hat-virtualization.

[10] Seagate: What is a 3-2-1 backup strategy? (2024. április 7.). https://www.seagate.com/gb/en/blog/what-is-a-3-2-1-backup-strategy/.

[11] SUSE: SUSE Certifications and Features (2024. április 24.). https://www.suse.com/support/security/certifications/.

[12] SUSE: SUSE Linux Enterprise Server 15 SP5 – Virtualization Guide (2024. április 11.). https://documentation.suse.com/sles/15-SP5/pdf/book-virtualization_en.pdf.

[13] SUSE: SUSE Linux Enterprise Server Documentation: Managing Networks (2024. május 6.). https://documentation.suse.com/sles/15-SP2/html/SLES-all/cha-libvirt-networks.html.

[14] Wikipedia: Standard raid levels (2024. április 7.). https://en.wikipedia.org/wiki/Standard_RAID_levels.

[15] XEN Project: Xen project 4.4 release notes (2024. március 20.). https://wiki.xenproject.org/wiki/Xen_Project_4.4_Release_Notes.

[16] Yev Pusin, Backblaze: The 3-2-1 backup strategy (2024. április 7.). https://www.backblaze.com/blog/the-3-2-1-backup-strategy/.