ktu
1922

Virtualios infrastruktūros sauga

Konteineriai

T120M144

2020

Turinys

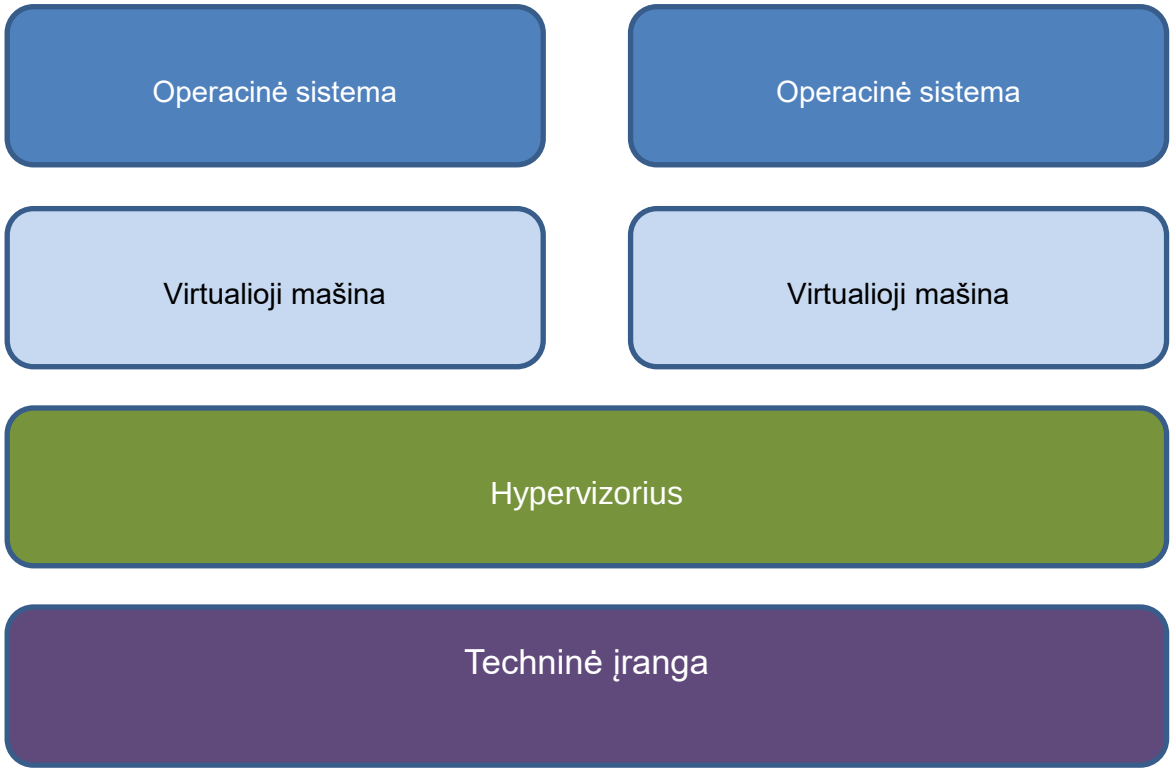
- **Virtualizacija ir jos taikymai**
- **Konteineriai (Docker)**
- **Laboratorinis darbas**

Virtualizacija

virtualizacija šaknis – žodis virtualus (lot. *virtualis*), kuris tarptautinių žodžių žodyne paaiškinamas kaip galimas, tariamas; galintis arba turintis pasireikšti tam tikromis sąlygomis; perskaičiuotas kitokioms sąlygoms, negu tiriamosios.

Virtualizacija yra karkasas arba metodologija kompiuterio ištekliams padalinti į keletą vykdymo aplinkų, taikant vieną ar daugiau principų ar technologijų, pavyzdžiui, techninės ir programinės įrangos suskaidymo, laiko padalijimo, dalinio ar visiško mašinos modeliavimo, emuliacijos, paslaugų kokybės užtikrinimo, ir kitų.

Schematiškai



Virtualizacija. Nauda (1)

Pateiksime kai kurias reikšmingas priežastis, skatinančias naudoti virtualizaciją, ir virtualizacijos teikiamą naudą:

Virtualios mašinos gali būti naudojamos siekiant kelis nepakankamai apkrautus serverius apjungti į mažesnį mašinų skaičių (galbūt, vieną mašiną); tai vadinama serverių konsolidavimu. Tokiu būdu yra taupoma aparatūra, serverių infrastruktūros valdymo ir administravimo išlaidos, išlaidos aplinkos apsaugai.

Virtualios mašinos gerai tinka senoms taikomosioms programoms vykdyti, kurios gali neveikti naujesniuose kompiuteriuose ir / ar operacinėse sistemose. Net jei ir veiktų, jos gali nepakankamai apkrauti serverį, todėl, kaip nurodyta aukščiau, prasminga konsoliduoti kelis taikymus.

Virtualizacija. Nauda (2)

Virtualios mašinos gali būti naudojamos kuriant saugias, izoliuotas „smėlio dėžes“ (sandbox) nepatikimoms programoms vykdyti. Galima netgi sukurti tokią aplinką dinamiškai, kai reikia vykdyti atsisiųstą iš interneto programą. Virtualizacija yra svarbus principas kuriant saugias skaičiavimo platformas.

Virtualios mašinos gali būti naudojamos siekiant sukurti operacines sistemas ar vykdymo aplinkas esant ribotiems ištekliams.

Virtualios mašinos gali suteikti aparatūros arba aparatūros konfigūracijos, kurios faktiškai neturite, iliuziją (pvz., SCSI įtaisų, kelių procesorių ir pan.). Virtualizacija taip pat gali būti naudojami siekiant imituoti nepriklausomų kompiuterių tinklą.

Virtualios mašinos gali būti naudojamos norint paleisti kelias operacines sistemas vienu metu: naudoti skirtingas jų versijas ar net visiškai skirtingas sistemas, kurios gali būti karšto budėjimo režimu. Kai kurias tokias sistemas gali būti sunku arba neįmanoma paleisti naujesnėje techninėje įrangoje.

Virtualizacija. Nauda (3)

Virtualios mašinos įgalina naudotis galingomis derinimo ir našumo stebėsenos priemonėmis. Pavyzdžiui, tokias priemones galima įdėti į virtualios mašinos monitorių. Operacinės sistemos gali būti derinamos neprarandant našumo, ar tam panaudojami sudėtingesni derinimo scenarijai.

Virtualios mašinos gali izoliuoti vykdomas programas, taip izoliuojant galimus gedimus ir klaidas. Tuomet galima imituoti programinės įrangos klaidas ir studijuoti tolesnį jos elgesį.

Virtualių mašinų sudaro sąlygas lengviau programinės įrangos migracijai, taip padidinant programų ir sistemų mobilumą.

Taikomųjų programų rinkinį traktuojant kaip instrumentą, tkį jų paketą galima paleisti atskiroje virtualioje mašinoje.

Virtualios mašinos yra puikūs įrankiai moksliniams tyrimams ir akademiniams eksperimentams. Kadangi jos gali būti izoliuotos, su jomis saugiau dirbti. Jos inkapsuliuoja viską sistemos būseną: galite išsaugoti būseną, ją išnagrinėti, pakeisti, perkrauti ir pan.

Virtualizacija. Nauda (4)

Virtualizacija gali įgalinti esamas operacines sistemas paleisti bendrosios atminties multiprocesoriuose.

Virtualios mašinos gali būti naudojamos siekiant sukurti įvairiausius bandymų scenarijus, siekiant veiksmingai užtikrinti sistemos kokybę.

Virtualizacija gali būti naudojama palyginus mažomis pastangomis pridedant naujas funkcijas esamose operacinėse sistemos.

Virtualizacija įgalina lengviau atlikti tokias užduotis, kaip sistemos migracija, atsarginių kopijų darymas ir sistemos atkūrimas.

Virtualizacija gali būti veiksminga priemonė realizuojant dvejetainį suderinamumą (binary compatibility).

Virtualizacija tinka saugiai ir ekonomiškai prieglobos funkcijų realizacijai.

Virtualizacijos lygiai

Virtualizacija gali būti vykdoma keliuose abstrakcijos lygiuose (Nanda, 2005):

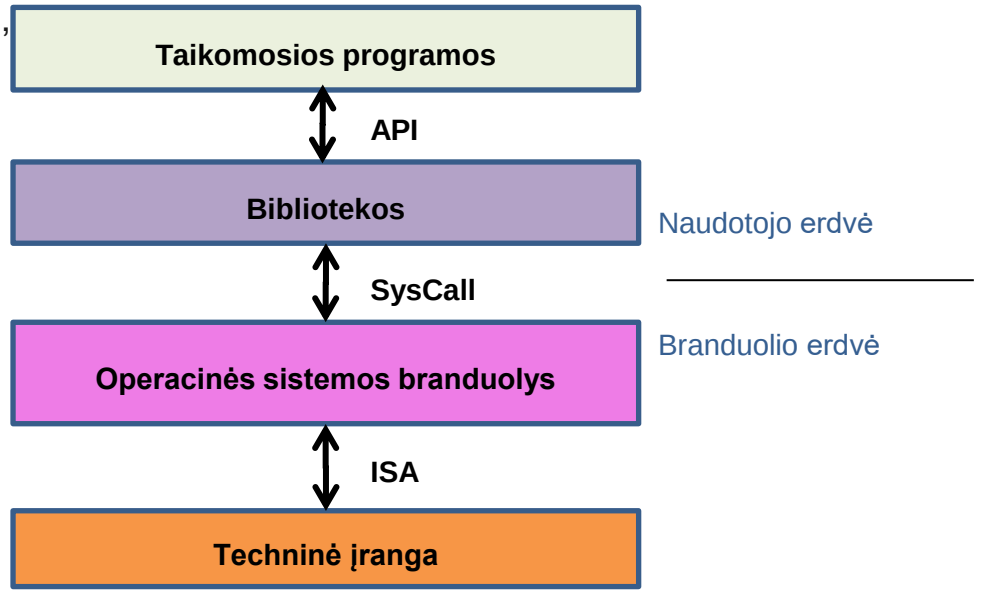
komandų sistemos (*ISA*) lygmenyje,

abstrakčiajame aparatūros lygmenyje (*Hardware Abstraction Layer* - HAL),

operacinės sistemos lygmenyje (sistemos iškvietos – *SysCall* – sąsaja),

naudotojo lygmens bibliotekos sąsajos,

taikomųjų programų lygmenyje .



Virtualizacijos lygiai. Komandų lygmuo

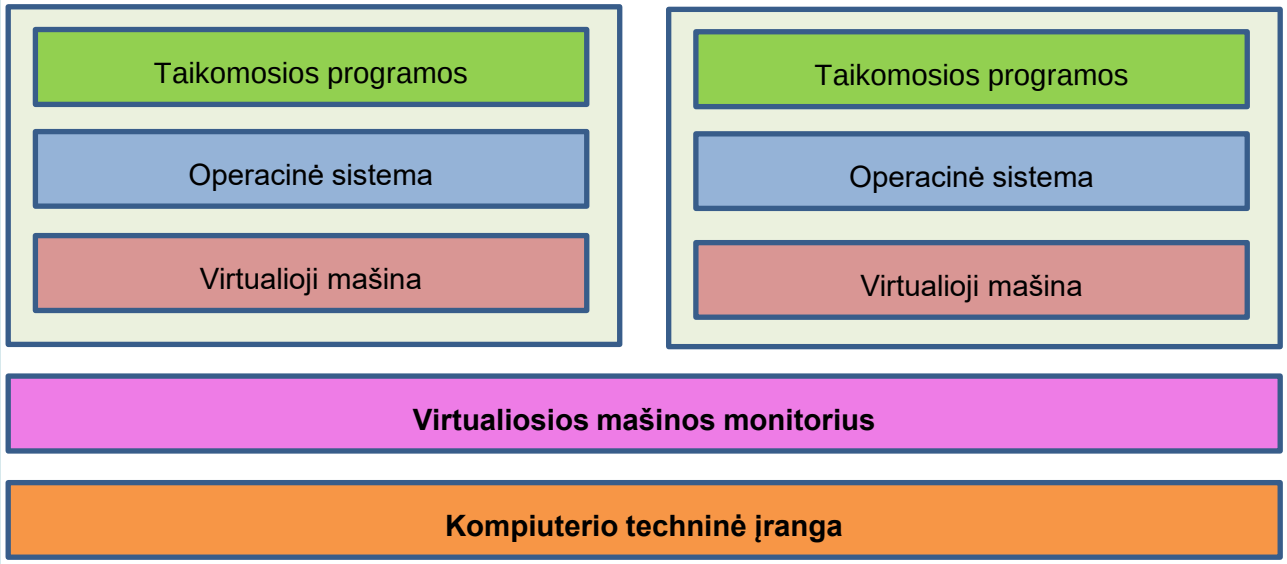
Virtualizacija komandų sistemos lygmenyje realizuojama programiškai emuliuojant komandų sistemos architektūrą. Tipišką kompiuterį sudaro procesoriai, atminties moduliai, magistralės, standieji diskai, diskų kontroleriai, įvairūs įvesties ir išvesties įtaisai ir pan. Emuliatorius bando vykdyti imituojamo kompiuterio išduodamas komandas, transformuodamas jas į savo komandų rinkinį ir tada jas vykdydamas turimoje aparatinėje įrangoje. Kad emuliatorius galėtų sėkmingai imituoti realų kompiuterį, jis turi sugebėti emuliuoti viską – įjungimą, paleidimą iš naujo ir t.t.

Virtualizacijos lygiai. Komandų lygmuo

Virtualizacija komandų sistemos lygmenyje realizuojama programiškai emuliuojant komandų sistemos architektūrą. Tipišką kompiuterį sudaro procesoriai, atminties moduliai, magistralės, standieji diskai, diskų kontroleriai, įvairūs įvesties ir išvesties įtaisai ir pan. Emuliatorius bando vykdyti imituojamo kompiuterio išduodamas komandas, transformuodamas jas į savo komandų rinkinį ir tada jas vykdydamas turimoje aparatinėje įrangoje. Kad emuliatorius galėtų sėkmingai imituoti realų kompiuterį, jis turi sugebėti emuliuoti viską – įjungimą, paleidimą iš naujo ir t.t.

Virtualizacijos abstrakčiam aparatūros lygmenyje

Išnaudoja emuliuojančios ir imituojamos platformų architektūros panašumus, kaip sumažinant emuliavimo įnešamus laiko nuostolius. Dauguma šiandien sutinkamų komercinių PC emuliatorių šį virtualizacijos metodą naudoja populiariose x86 platformose. Virtualizacija metodas padeda virtualius išteklius atvaizduoti realiuose fiziniuose ištekliuose ir atlikti skaičiavimus emuliuojančioje mašinoje.



OS lygmenyje

Virtualizacija operacinės sistemos lygmenyje panaudoja turimos platformos techninę ir programinę įrangą, o virtualizacijos sluoksnis (panašus į VMM) kuriamas virš naudojamos operacinės sistemos, kad naudotojui būtų pateikiamos kelios nepriklausomos ir izoliuotos mašinos.

Kalbos lygmenyje

Virtualizacijos ***programavimo kalbos lygmenyje*** idėja – sukurti virtualią mašiną taikomosios programos lygyje. Ji palaiko naują apibrėžtą komandų (Java baitų kodai JVM) rinkinį. Tokie VM kelia nedidelę grėsmę sistemos saugumui, kadangi naudotojui leidžiama žaisti su ja paleidžiant programas, kaip kad jis darytų fizinėje mašinoje.

Beveik visose sistemose taikomosios programos rašomos naudojant taikomųjų programų kūrimo sąsajas (API – *Application Programming Interface*), eksportuojamas naudojant naudotojo lygmens bibliotekas. Tokios bibliotekos skirtos su operacinė sistema susijusioms realizacijos detalėms paslėpti, kad eilinių programuotojų darbas būtų paprastesni1s. Tokia ***virtualizacija bibliotekų lygmenyje*** suteikia virtualizacijos bendruomenei naujas galimybes.

Virtualizacijos lygmenų palyginimas

	Komandų sistemos lygmuo	Abstraktusis aparatūros lygmuo	Operacinės sistemos lygmuo	Programavimo kalbos lygmuo	Bibliotekų lygmuo
Našumas	+	++++	++++	++	+++
Lankstumas	++++	+++	++	++	++
Realizacijos paprastumas	++	+	+++	++	++
Izoliavimo laipsnis	+++	++++	++	+++	++

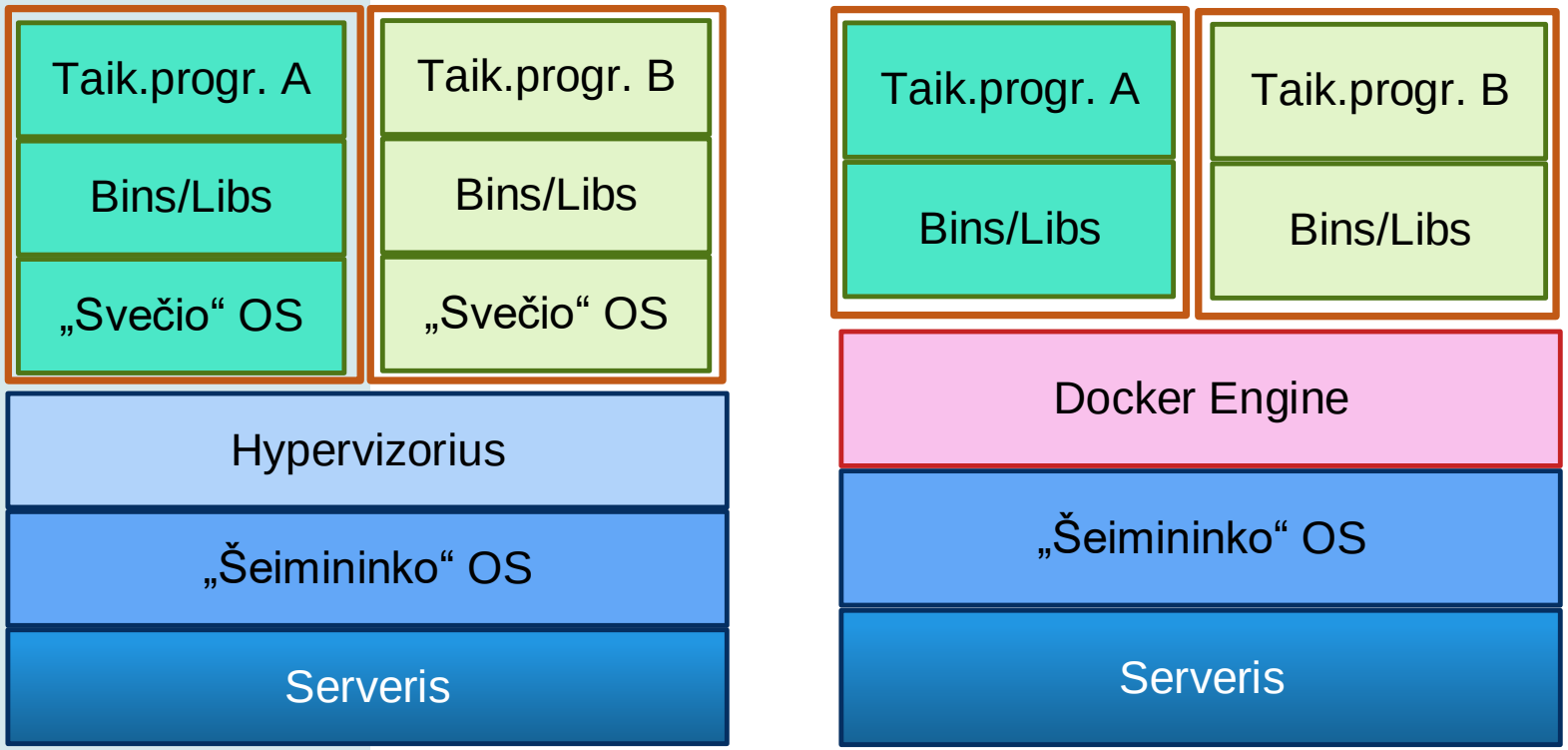
Konteineriai

Virtualizacija šiandien yra gerai suprantamas požiūris bendrai naudojant serverio išteklius, tačiau yra keletas našumo ir išteklių panaudojimo efektyvumo problemų, susijusių su virtualizacija hypervizoriaus pagrindu. Neseniai pasirodė naujas būdas, pavadintas *virtualizacija naudojant konteinerius*, kuris padada spręsti šias problemas.

Virtualus konteineris iš esmės panašus į tikrą konteinerį – jame „supakuojama“ viena taikomoji programa bei jos bibliotekos ir jai reikalinga operacinė sistema. Toks konteineris būna izoliuotas nuo kitos programinės įrangos, o jame esanti taikomoji programa gali būti lengvai perkeliama iš vienos mašinos į kitą.

Virtualios mašinos ir Konteineriai

Virtualiųjų mašinų (kairėje) ir Docker debesų konteinerių (dešinėje) palyginimas



Hypervizorius ir Konteineriai

Virtualiųjų mašinų hipervizorių ir konteinerių palyginimas

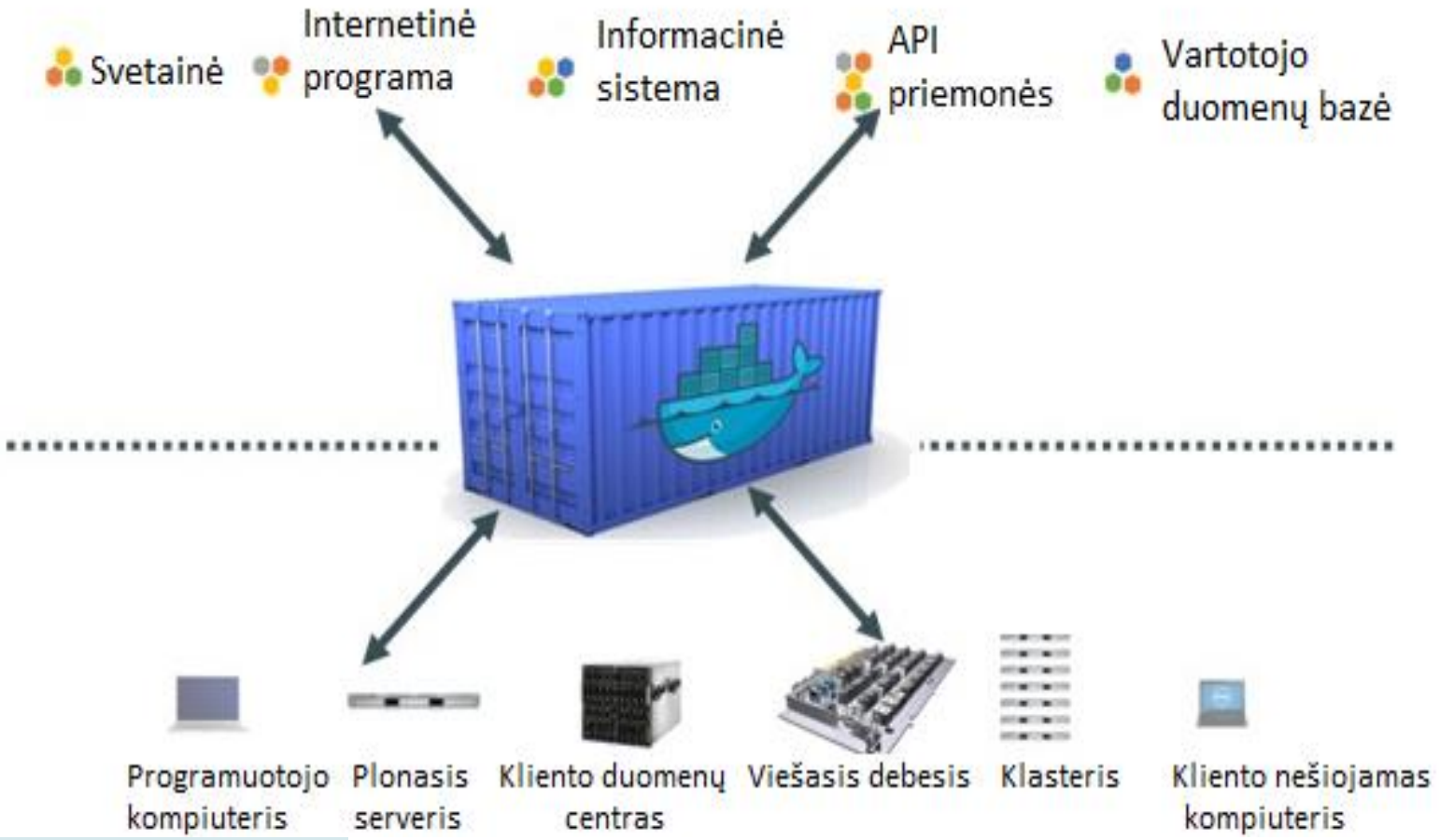
Savybė	Hipervizoriai	Konteineriai
Virtualizacijos lygmuo	Techninės įrangos lygmuo	Operacinės sistemos lygmuo
Kas abstrahuojama	Taikomoji programa nuo operacinės sistemos	Operacinė sistema nuo techninės įrangos
Atminties naudojimas	Kiekvienam objektui sunaudojama apibrėžta atminties sritis	Naudoja vieną bendrą atminties sritį plius mažas sritis kiekvienam sluoksniui
Paleidimo sparta	Įkrovos laikas atitinka įprastinės OS paleidimo laiką, priklausantį nuo išorinės atminties spartos (apie 20 sekundžių)	Gali būti paleisti ir pasirengę vykdyti taikomąją programą mažiau nei per 500 ms

Docker

Virtualūs konteineriai pradėjo sparčiai populiarėti pastaruoju metu, ypač pasirodžius „Google“ plėtojamai „Docker“ platformai, kuri leidžia automatizuoti virtualių konteinerių kūrimo procesą.

Docker konteineriai remiasi Linux technologija, pavadinta *cgroups* (*control groups* santrumpa) – Linux branduolio savybe paskirstyti ir izoliuoti įvairių išteklių (procesoriaus, pagrindinės atminties, diskų ir pan.) panaudojimą realizuojant procesų rinkinį. Docker – priemonė, kuri automatizuoja taikomosios programos paleidimo procesą, jos kodą supakuodama į konteinerį, kuris gali būti paleistas vykdyti kaip programa beveik bet kokioje kompiuterinėje aplinkoje. Docker kūrėjai tvirtina, kad ši priemonė gali inkapsuliuoti bet kokį kodą ir paleisti jį bet kokiame serveryje. Programuotojas, išbandęs sukurtą konteinerį savo stacionariame kompiuteryje, gali po to jį paleisti virtualioje mašinoje, serveryje, klasteryje ir pan.

Docker panaudojimas



Docker

Kaip pažymi Docker kūrėjai (The whole story, 2013), ši technologija eliminuoja tai, kas vadinama „priklausomybės pragaru“ (*dependency hell*) – Docker pašalina painiavą, kylančią esant dideliam komponentų ir priklausomybių tarp jų skaičiui. Įsivaizduokite, kad jūs turite daug programinės įrangos komponentų: naudotojo duomenų bazę, informacinę sistemą, internetines programas, API priemones, statinio kodo interneto svetainę ir pan. Visi šie tarpusavyje susiję komponentai turi būti sutalpinti „geležies“ komponentuose, įskaitant virtualias mašinas, plonąjį serverį, duomenų centrą, viešąjį debesį, klasterį, naudotojų kompiuterius.

Docker, kaip žinia, yra sukurtas ant Linux pamatų; ši atviro kodo operacinė sistema jau nustelbė Windows tarp interneto gigantų. Kad Windows išliktų konkurencingi, Microsoft turi pasiūlyti savo operacinėje sistemoje kažką panašaus į Docker, todėl bendrovė ketina konteinerių technologijas panaudoti naujoje Windows Server operacinėje sistemoje, kad paleisti dideles interneto paslaugas pasaulio duomenų centruose.

Laboratorinis

Priemonės

- Linux (Debian 8/9 Ubuntu 17.04)
- Docker atvaizdai (images)
- HTTP
- DB (MySQL)

Darbų eiga

Sistemos reikalavimai.

Docker alterway/php 5.4-apache id - b4010e5adb26 (apache 2.4.7-1ubuntu4.15, php 5.5.9+dfsg-1ubuntu4.21);

Docker mysql id - e799c7f9ae9c (mysql-client 5.7.18-1debian8, mysql-server 5.7.18-1debian8)

Diegimas

1. Įdiegiame Linux’ine mašiną.
2. Prisijungiame kaip "root" naudotojas

2.1. sudo -i

2.2. Įvedame "root" naudotojo slaptažodį

3. Įdiegiame atnaujinimus: apt-get update
4. įdiegiame net-tools'us: apt-get install net-tools
5. įdiegiame Docker'į: apt-get install docker.io (jei Debian 8 docker.ce)

Diegimas (2)

6. parsisiunčiam konteinerį apache+php5- nimmis/apache-php5: docker pull nimmis/apache-php5
7. parsisiunčiam konteinerį mysql: docker pull mysql
8. Patikriname ar repozitorija sukūrta: docker images
9. parsiunčiame docker-compose: apt-get install docker-compose

(jei Debian 8

sudo apt-get install python-pip -y

sudo pip install --upgrade pip

sudo pip install docker-compose

docker-compose –version)

Diegimas (3)

10. Sukūrti failą docker-compose.yml: cd /opt/

10.1. vim docker-compose.yml

10.2. Įterpti informaciją, kur bus:

Aprašytas image: nimmis/apache-php5

Aprašytas image: mysql

- ports:80
- links
- volumes
- environment

Diegimas (4)

11. bandome paleisti konteinerius būdami /opt/ direktorijoje (cd /opt/)
12. žiūrime ar konteineriai paleisti
13. Bandome prisijungti prie paleisto konteinerio "opt_php-5.2_1” ("opt_php-5.2_1"<- Name, kurį galime rasti parašius komandą)
14. Bandome prisijungti prie paleisto konteinerio "opt_mysql_1" ("opt_mysql_1" <- Name, kurį galime rasti parašius komandą)

Diegimas (4)

14.1. Prisijungti prie duomenų bazės

14.1.1. Žiūrėti kokios lentelės: SHOW DATABASES; <-- komanda, jeigu reikia.

14.1.2. Sukurti naują lentelę: CREATE DATABASE pavadinimas;

15. Pažiūrėti kokį IP adresą gavo konteineris

16. Paleidžiame naršyklę linux mašinoje ir pažiūrime ar veikia mūsų web'as.

17. Įdiegti mysql klientą pagrindinėje mašinoje

18. Einame į www direktoriją būdami savo pagrindinėje mašinoje, o ne konteineryje

Papildomai. Valdymas

Papildomos komandos:

docker stop konteinerio_vardas

docker rm -f konteinerio_vardas

https://docs.docker.com

Laboratorinio mašinos parametrai

NAME="Ubuntu"

VERSION="17.04 (Zesty Zapus)"

ID=ubuntu

ID_LIKE=debian

PRETTY_NAME="Ubuntu 17.04"

VERSION_ID="17.04"

HOME_URL="https://www.ubuntu.com/"

SUPPORT_URL="https://help.ubuntu.com/"

BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"

PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"

VERSION_CODENAME=zesty

UBUNTU_CODENAME=zesty

ktu
1922

sarunas.grigaliunas@ktu.lt