Veebiteenuste ja hajussüsteemide arendus

Loeng 8: Virtualiseerimine ja konteinerid

Pelle Jakovits

jakovits@ut.ee

Kevad 2025

Loengu kava

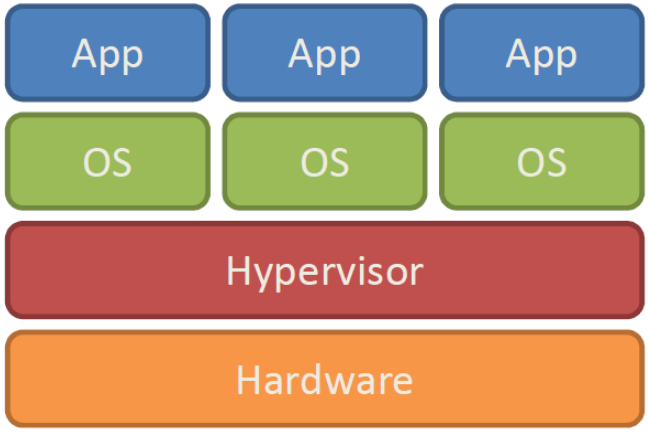
- Virtualiseerimine
  - Virtualiseerimine vs Konteinerid
- Konteinerite tehnoloogia
  - Nimeruumid
  - Cgroups
  - Failisüsteemid
- Konteinerite standardid
- Eelised ja väljakutsed

Mis on Virtualiseerimine

- Virtuaalse serveri loomine, mis suudab jooksutada oma operatsioonisüsteemi ja selle poolt toetatud rakendusi
- Virtuaalsete ressursside loomine
  - Failisüsteem, CPU, RAM, Võrgukaart, jne.
  - Ühe riistvara seadme ressursside jagamine mitme virtuaalse vahel
- Võimaldab pakkuda rakenduste jaoks turvalist, kohandatavat ning isoleeritud keskkonda
  - Linuksi jooksutamine Windows OS'ga arvutis
  - Ühes VM's jooksev rakendus ei saa mõjutada teises VM's asuvat rakendust

Virtualiseerimine

- Virtualiseerimise tehnikad on pilvetehnoloogia aluseks
- Virtualiseerimine võimaldab jaotada riistvara alamosadeks
  - Võimaldab pakkuda paindlikku ja skaleeritavat ligipääsu arvutusressurssidele
  - Virtuaalmasinate loomine
  - Virtuaalsed kettad, võrgud, failisüsteemid, keskkonnad, jne.
- Virtuaalmasinate loomise lahendused
  - VMware
  - Xen
  - KVM



Virtualized Stack

Motivatsioon

- Alakasutatud riist- ja tarkvararessursid
  - Serverist piisab tavaliselt rohkema kui ühe rakenduse jooksutamiseks
- Lihtsustab keskkondade kohandamist rakenduste jaoks
  - Igal rakendusel oma individuaalne keskkond
- Keskkondade ja rakenduste teisaldatavus ja taaskasutatavus
  - Tihti piisab failisüsteemi koopia ja süsteemi konfiguratsiooni liigutamisest
- Lihtsustab haldust
  - Riistvara jälgimine, defektse riistvara asendamine
  - Serveri seadistamine ja värskendused
  - Süsteemide varukoopiate tegemine
- Turvalisus
  - Rakenduste isoleerimine

Virtualiseerimise osapooled

- **Võõrustav masin** – Host Server
  - Masin, mis pakub teenust virtuaalmasinate jooksutamist
  - Olenevalt tüübist võib omada oma OS'I või jooksutada virtualiseerimise tarkvara otse riistvara peal
- **Külalis-masin** – Guest VM, Guest OS.
  - Virtuaalmasin, mis jookseb võõrustava masina sees. On oma operatsioonisüsteem.
- **Hüperviisor** - Hypervisor
  - Tarkvara kiht, mis võimaldab virtuaalmasinaid luua ning neid hallata
  - Hoolitseb võõrustava masina ressursside jagamise eest VM'de vahel
  - Monitoorib virtuaalmasinaid

Hüperviisor (HV)

- Tarkvara, mis võimaldab samaaegselt käivitada mitut VM'i ühe s füüsilises serveris
- Hüperviisor (HV) töötab juhendaja (Privilegeeritud) režiimi koha
- Loob virtuaalseid seadmeid ja keskkondasid
- Kaks peamist tüüpi:
  - **Tüüp I** – Otse riistvaral töötav HV. Asendab host OS'i.
  - **Tüüp II** - Operatsioonisüsteemi sees rakendusena töötav HV.

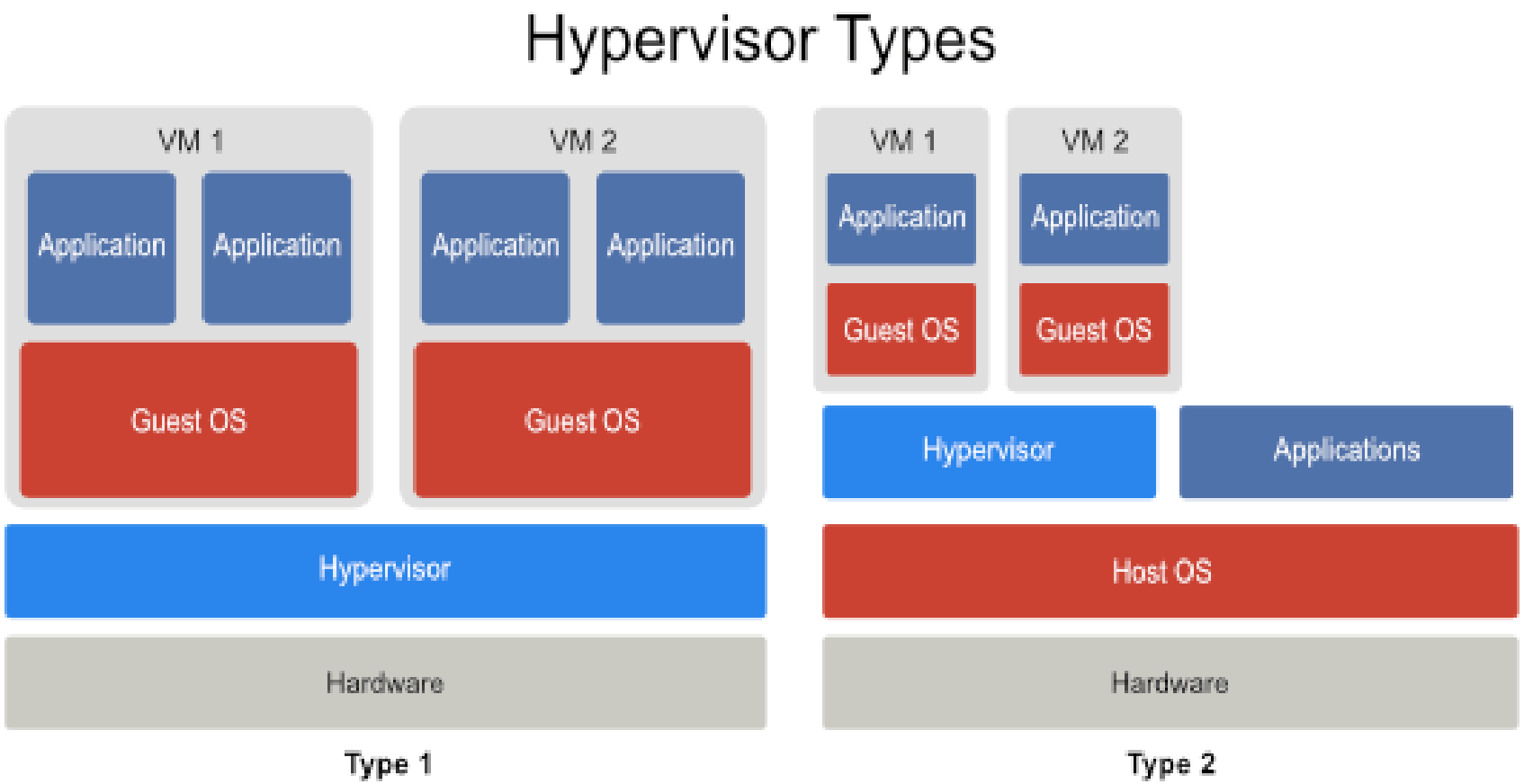
I Tüüp

- Töötab otse riistvara peal
  - Asendab Operatsioonisüsteemi
  - Pakub põhilise operatsioonisüsteemi funktsionaalsuseid
- Suhtleb otse riistvara poolt pakutud ISA-ga
- Tuntud ka kui põline (native) virtuaalmasin.
- Näited: Xen, VmWare ESXi, MS HyperV, KVM

II Tüüp

- Vajab OS tuge virtualiseerimise teenuse pakkumiseks
  - Virtualiseerija on OS-I poolt hallatav programm
  - Vajab kõrgemaid õigusi kui tavaprogrammid
- Võimaldab virtualiseerida teise arhitektuuriga süsteeme
- Seda nimetatakse ka hostitud virtuaalmasinaks
- Näited:
  - VirtualBox
  - KVM (Muudab Linux'i hüperviisoriks)

HV tüüpide võrdlus



https://www.vembu.com/blog/type-1-and-type-2-hypervisor/

Virtualiseerimise tehnikad

- **Täielik virtualiseerimine**
  - Võimalus käivitada virtuaalmasinaid ilma VM sisu muutmata
  - Vajab CPU tuge (kõike ei toeta)
  - Külalis-VM ei tea et see on virtuaalmasin
  - Aeglasem, osa CPU käske tõlgitakse
- **Para-virtualiseerimine – Kõrval-virtualiseerimine**
  - Modifitseeritud VM OS'd - Nõuab tarkvara lisamist VM sisse
  - Keerulisemate käskude täitmine otse host-masinal
  - VM'd saavad piiratud ligipääsu otse riistvarale (effektiivsem)
- **Riistvara-toetatud virtualiseerimine**
  - Riistvara pakub arhitektuurilist tuge VM-i ehitamiseks, mis suudab tööle panna külalisOS-i täielikus isolatsioonis
  - Intel Virtualization Technology (VT-x) and AMD's AMD-V
  - Isoleeritud järjekorrad VM'de jaoks riistvara kontrollerite sees
  - Spetsiaalsed riistvara komponendid virtuaalmasinate jaoks
    - Nt. **Translation lookaside buffer** (**TLB**)

Virtualiseerimise puudused

- Jõudluse halvenemine
  - Lisab uue vahevara/kihi host ja guest süsteemi vahel
  - Rakenduse käivitamine VM sees on aeglasem
    - Oleneb virtualiseerimise tüübist
  - Iga isoleeritud rakenduse jaoks on vaja oma OS koopiat
- Virtualiseerimisega seotud turvaaugud ja uued ohud
  - Näide: Virtualiseerimise puhul on võimalik, et pahatahtlikud programmid saavad end tööle panna enne, kui süsteemi OS käima pannakse.

Konteinerid

- Iga rakenduse jaoks ei ole mõtet üles seada oma VM'I ja OS'I
  - Väikesed rakendused
  - Mikroteenused
- Konteinerid aitavad lihtsamini eraldada rakendused üksteisest
- Serveris on minimaalne Linuxi operatsioonisüsteem ning kõik muu saab kävidata mingis vormis konteinerites
- Monoliitsetest rakendustest liikumine hajutatud mikroteenuste põhistele rakendusele
- Kiirem prototüüpimine ja juurutamine (CI/CD)
- Server vajab minimaalset Linuxi operatsioonisüsteemi ja kõik muu jookseb konteinerites

Konteinerid

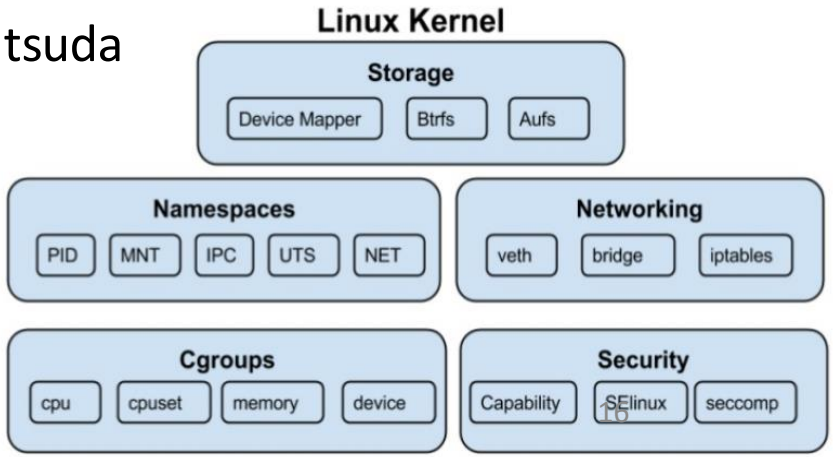
- On kergekaalulisemad
  - Saab otse kasutada OS'I kernelit
    - Ei ole vaja eraldi hüperviisorit (hypervisor)
  - Konteineri sisesed protsessid on eraldatud teistest konteineritest kasutades Linuksi nimeruume
  - Igal konteineril on komplekt talle eraldatud ressurssidest
- Kiire ülesseadmine
  - Konteineri alustamine on kiirem kui VM alustamine
- Jõudlus
  - Ligilähedane serveri tavajõudlusele, parem kui VM puhul
- Erinevad konteinerite raamistikud
  - LXC, Docker, Linux VServer, OpenVZ
  - Docker on avatud platvorm

Konteinerite implementeerimine

- LXC – **L**inu**X** **C**ontainers
- LXC on Linuxi Kerneli tasemel virtualiseerimismeetod mitme isoleeritud linuxipõhise süsteemi jooksutamisel ühel hostil.
- Kasutame süsteemide eraldamiseks Linux nimeruume ja Cgruppe (Cgroup).
  - **Nimeruumid**: Protsesside, failisüsteemide grupeerimine üksteisest eraldatud nimeruumidesse
  - **Cgroups**: Nimeruumidele CPU, RAM, kettaruumi jms. ressursside eraldamine ja mahu piiramine.

Konteinerite komponendid

- Konteinerite **pildid** (**image**) - Kombineerib rakenduse koodi, keskkonna (runtime) tarkvara ja kõik vajalikud teegid ühte käivitatavasse paketti
- Linuxi **nimeruumid** ja **C-groups** süsteemide eraldamiseks.
- **Kihilised failisüsteemid** paljudest allikatest pärit failide liitmiseks üheks konteinerile nähtavaks failisüsteemiks
- **SELinux** – Piiramine, millised välised failid, pordid, süsteemi teenused on konteineris kasutatavad
- **SECCOMP** – piirab, milliseid **süsteemi käske** (**syscall**) konteineri protsess võib välja kutsuda



Konteineri pilt (image)

- Konteinerpilt on käivitamiseks valmis pakitud tarkvara, mis sisaldab:
  - Rakenduse kood
  - Rakenduse käitamiseks vajalik töökeskkonna (runtime) tarkvara
    - Nt. Python 3.7
  - Vajalikud süsteemi ja töökeskkonna teegid
    - Nt. wget linuxi programm, Flask Pythoni teek
  - Vaikekonfiguratsiooni väärtused
    - Nt. Teenuse port, vaike kasutajanimed
  - Rakenduse käivitamise käsk (või skript)
- Konteinerpildist saab kergesti teha koopiaid uute konteinerite üles seadmiseks
- Konteineri pilt on ainult loetav
  - Muutmiseks tuleb luua uus pilt

Nimeruumid

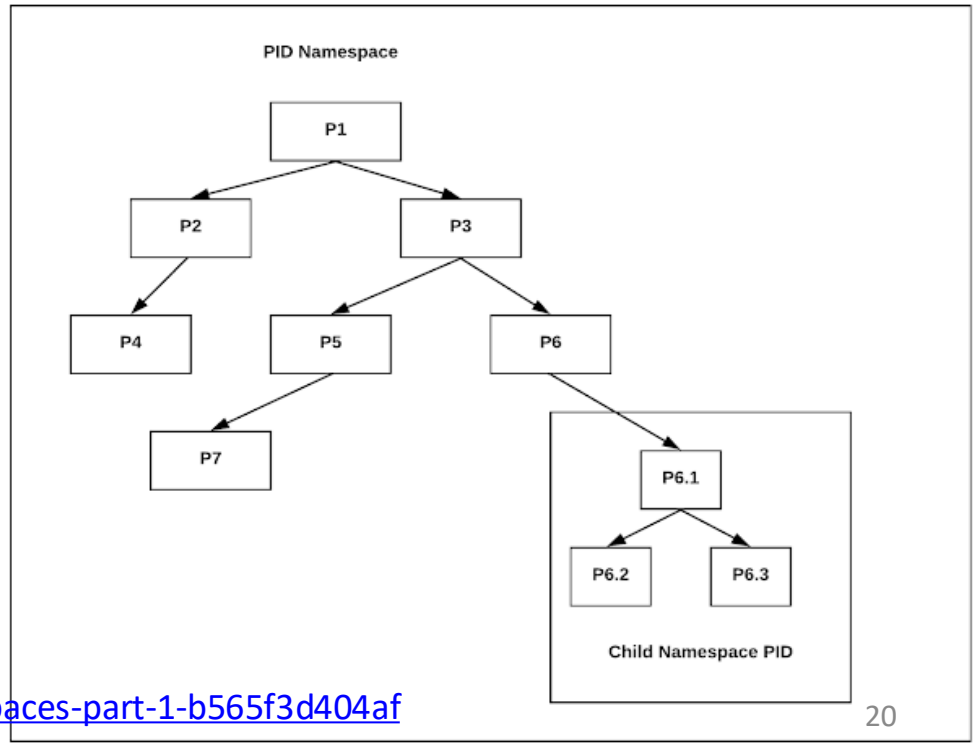
- Algselt IBMi poolt välja töötatud Linuxi protsesside ja süsteem i ressursside partitsioneerimine
- Nimeruumid eraldavad ressursid eraldi aadressiruumideks
  - Riia 16 **Tartus** vs Riia 16 **Tallinnas**
  - Protsess **16** konteineris A **!=** Protsess **16** konteineris B **=!** Protsess **16** OS-is
- Võimaldab ressursse isoleerida OS kerneli tasemel
  - Ainult samas nimeruumis olevad protsessid pääsevad juurde selle nim eruumi ressurssidele ja protsessidele

Nimeruumide tüübid

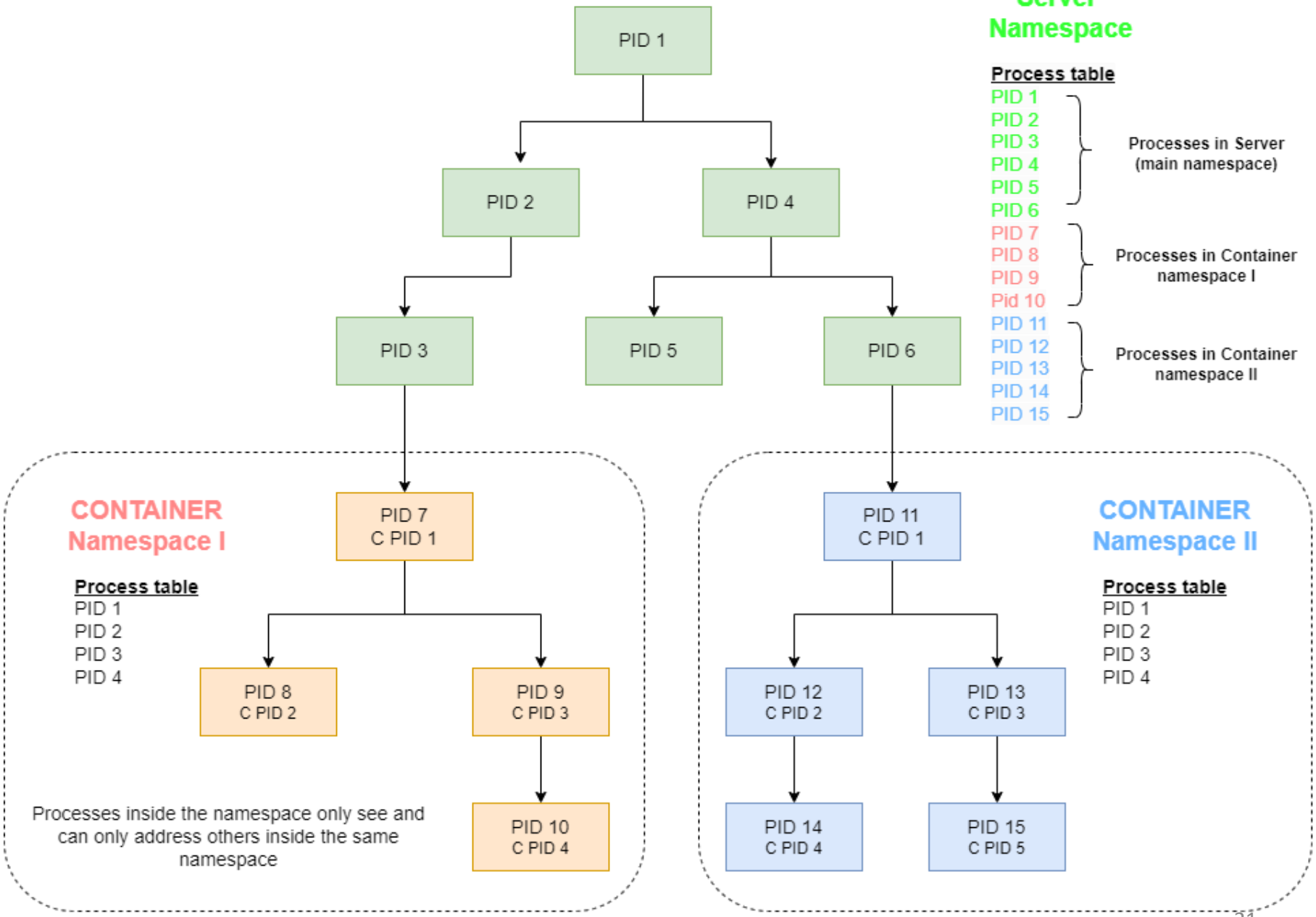
- Protsesside nimeruum (PID)
- Failisüsteemi nimeruum (MNT)
- Võrgu nimeruum - Võrgukaardid, marsruutimise tabelid.
- Domeenide nimeruum
- Kasutajate nimeruum
- IPC nimeruum - protsesside vahelise suhtluse eraldamiseks

PID Nimeruumid

- OS-il on üks protsessipuu ja protsessitabel
- Väiksemad nimeruumid näevad ainult väiksemat haru
- Kõik nimeruumis/harus loodud uued protsessid jäävad samasse nimeruumi
- Kuid need eksisteerivad ka globaalses nimeruumis
- Juurkasutaja (root) saaks konteineris oleva protsessi tappa

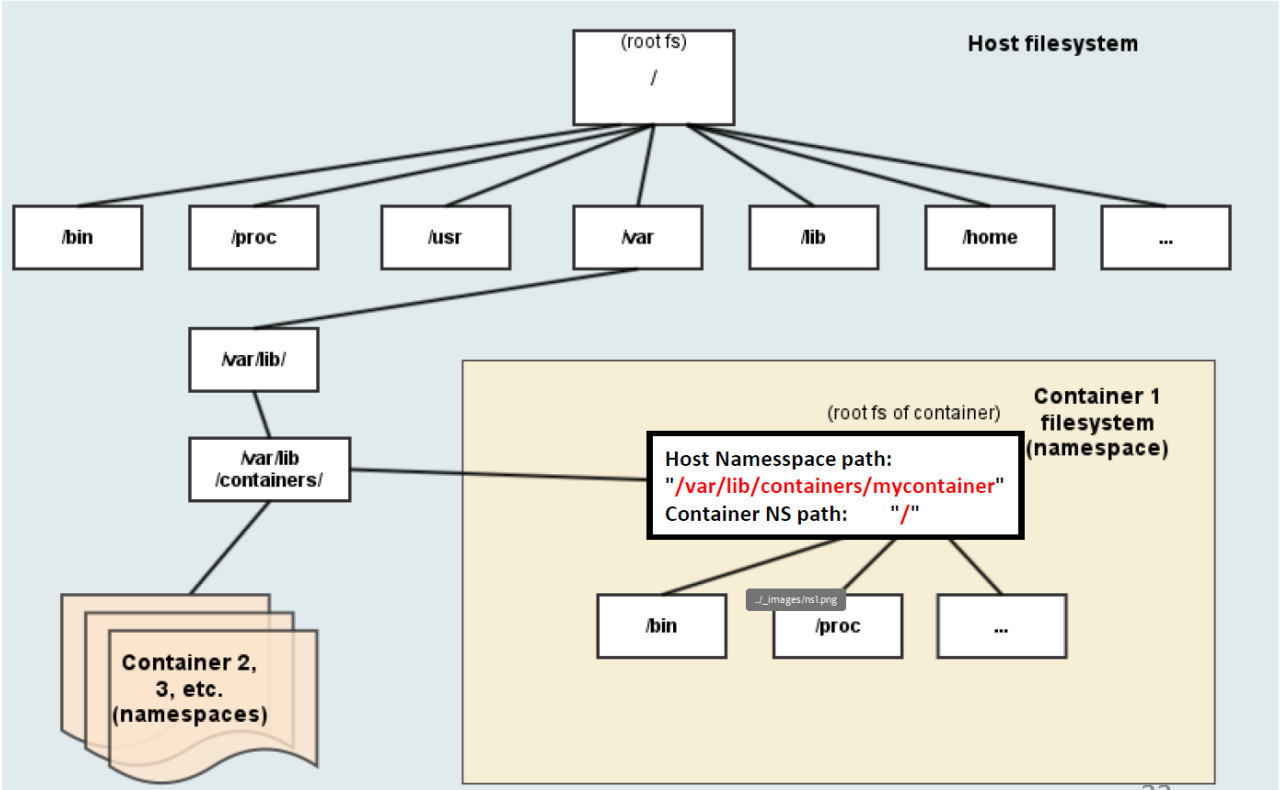


https://iamabhishek-dubey.medium.com/linux-namespaces-part-1-b565f3d404af


Server Namespace
Process table
PID 1
PID 2
PID 3
PID 4
PID 5
PID 6
PID 7
PID 8
PID 9
Pid 10
PID 11
PID 12
PID 13
PID 14
PID 15
Processes in Server (main namespace)
Processes in Container namespace I
Processes in Container namespace II
PID 1
PID 2
PID 4
PID 3
PID 5
PID 6
CONTAINER Namespace I
Process table
PID 1
PID 2
PID 3
PID 4
PID 7 C PID 1
PID 8 C PID 2
PID 9 C PID 3
PID 10 C PID 4
Processes inside the namespace only see and can only address others inside the same namespace
PID 11 C PID 1
PID 12 C PID 2
PID 13 C PID 3
PID 14 C PID 4
PID 15 C PID 5
CONTAINER Namespace II
Process table
PID 1
PID 2
PID 3
PID 4

MNT, Faili süsteemi nimeruumid

- OS-il on üks failisüsteemi nimeruum (Võib olla kihiline)
- Võib koosneda paljudest failisüsteemidest, mis on ühendatud (mounted) kindlatesse kaustadesse
- Üks "väline" kaust on konfigureeritud konteineri juurkaustaks
- Iga uus kaust või fail, mis on loodud nimeruumis jääb sellesse nimeruumi
- See eksisteerib endiselt failina välise OS-i failisüsteemis
- Root kasutaja saab konteineri nimeruumis oleva faili kustutada ka väljaspoolt

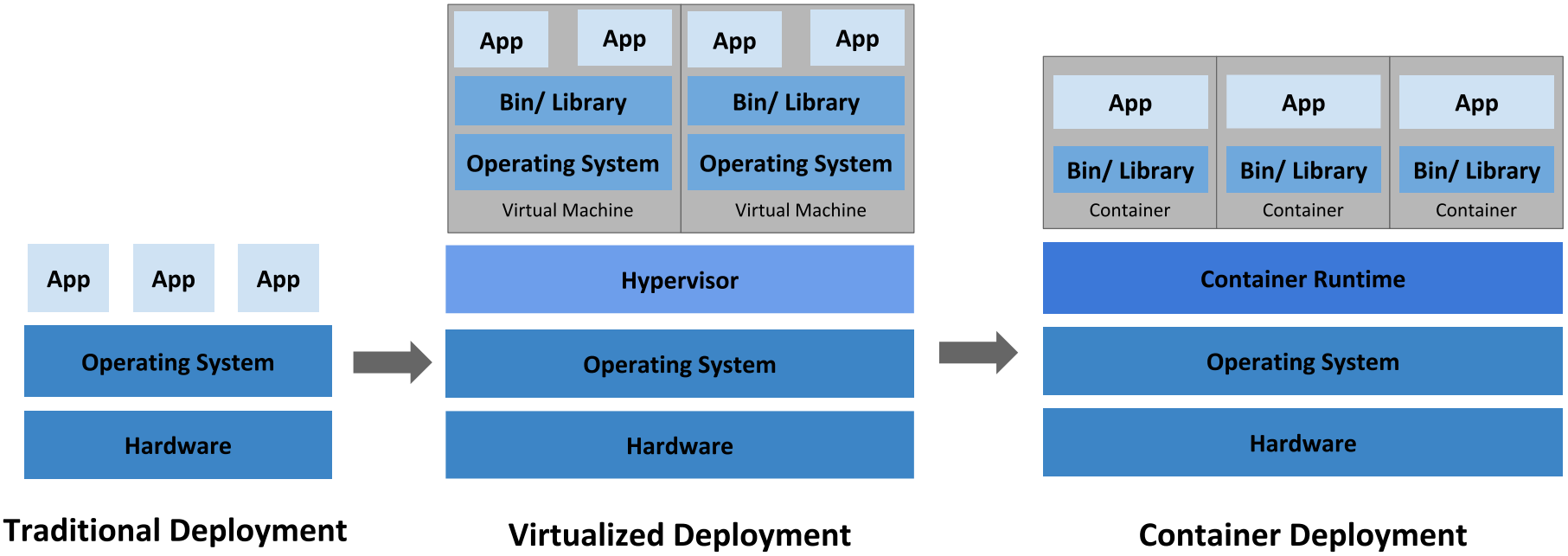


https://blog.selectel.com/containerization-mechanisms-namespaces/

Cgroups

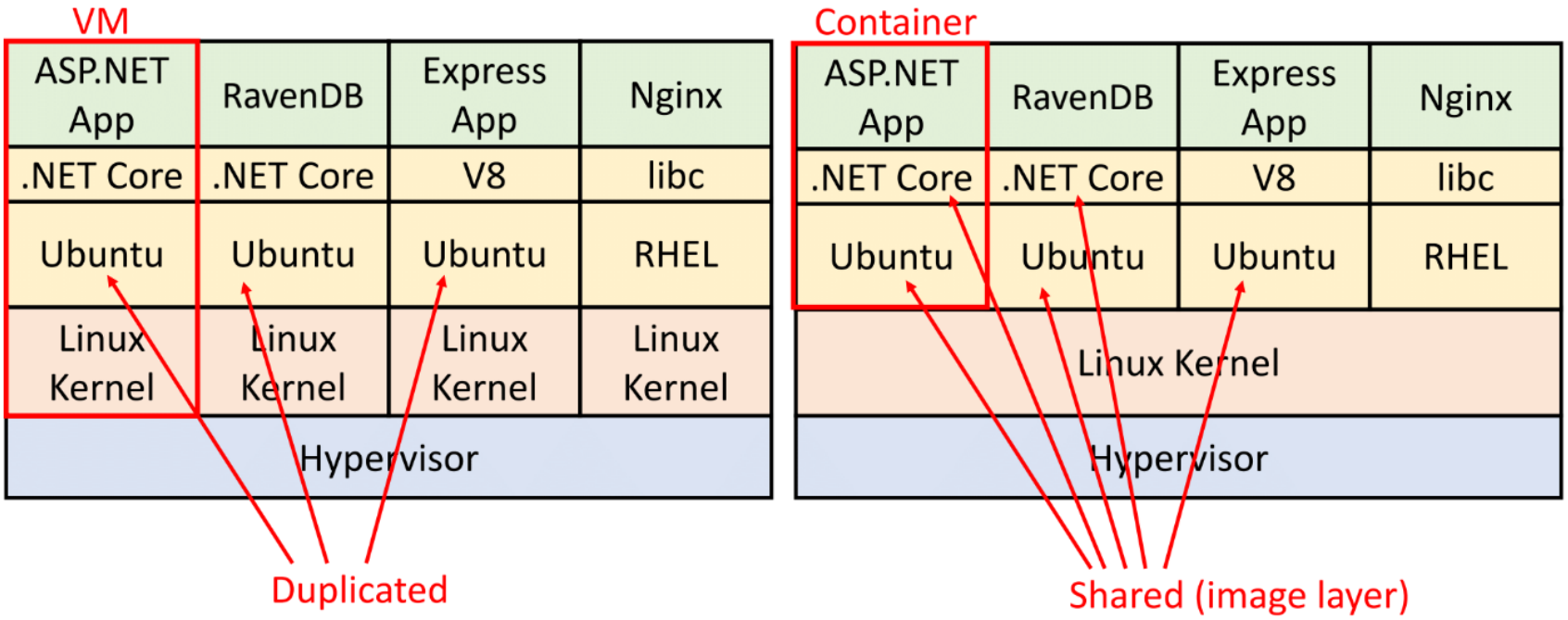
- Algselt Google'i poolt loodud Linuxi tuuma funktsionaalsus, sü steemiressursside kasutamise piiramiseks
- Lubada ressursside kasutamist ainult teatud nimeruumi protsesside poolt
  - Jagada CPU aega või Mälu mahtu nende vahel
- Näited
  - CPU aeg, ligipääs konkreetsetele CPU tuumadele
  - RAM maht
  - Võrgu liikluse kiirus/maht/prioriteet
  - Kettaruum
  - Ligipääs seadmetele

Virtualiseerimine vs konteineriseerimine



https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Virtualiseerimine vs konteineriseerimine



Sasha Goldshtein, #DotNext talk, https://twitter.com/goldshtn/status/988468555883696129

Docker

- 2013. aastal käivitas Solomon Hykes dotCloud pilves Dockeri asutuse sisese projektina.
- Docker avaldati avatud lähtekoodiga 2013 aasta märtsis
- Docker'i kommertsiaalne lahendus lasti välja 2016 aastal
- Docker'i kogukonna (community) väljaanne on tasuta ja toetatud 10k+ panustaja poolt.

Dockeri komponendid

- **Dockerfile** – Konfiguratsiooni fail, mis defineerib, kuidas konteinerit luua.
- Docker **container** – konteiner, mille sees jooksevad protsessid
- Docker **image** – konteineri ketta mall, mis on aluseks konteineri jaoks keskkonna loomiseks
- Docker **daemon** –
  - Protsess, mis haldab serveris jooksvaid konteinereid
  - Pakub API't Docker ressursside haldamiseks
- Docker **registry** – piltide register

Docker Pilt (Image)

- Dockeri konteineri alus - read-only mall konteineri loomiseks
- Docker kasutab union failisüsteemi.
- Dubleerimise vaba. Kastutab copy-on-write.
- Ehitatakse kihtidena
  1. Aluseks Ubuntu baas pilt
  2. Uue kihina installeerimine OpenJDK
  3. Uue kihina lisame JAR faili
  4. Uue kihina lisame konfiguratsiooni failid
  5. Viimases kihis paneme käima Java rakenduse

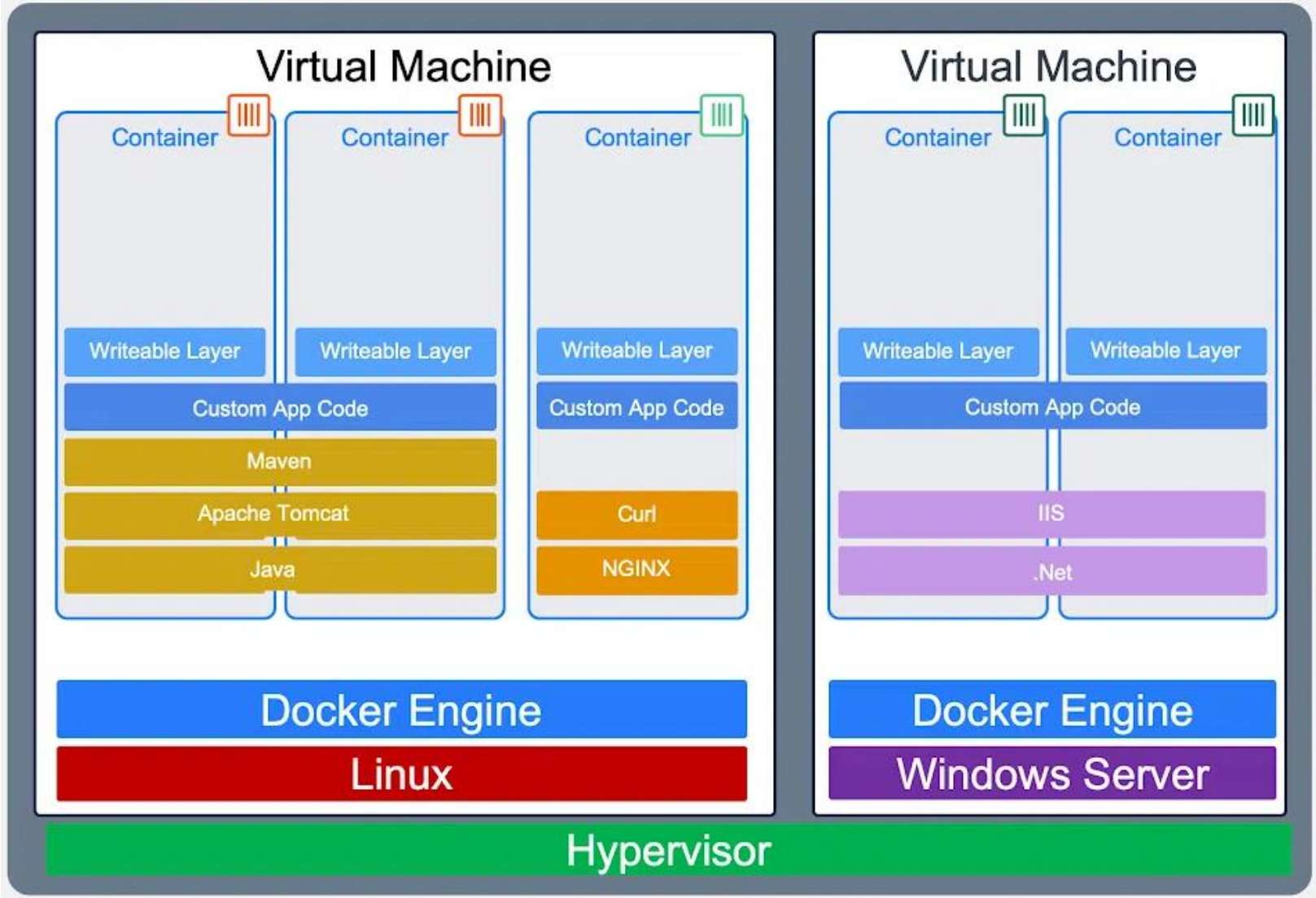
Copy-on-write

- Tehnika, mida kasutatakse ressursside jagamiseks protsesside vahel
- Välditakse ressursside kopeerimist nii kaua kui võimalik
- Mitu protsessi kasutavad sama read-only koopiat failidest/andmetest
- Alles siis kui protsess andmeid muudab, tehakse koopia ning see protsess kasutab tulevikus seda
  - Ülejäänud protsessid kasutavad muutmata kirjutuskaitstud koopiat

Union failisüsteem

- Linux failisüsteemi tüüp/implementatsioon
- Võimaldab erinevatest failisüsteemidest pärit faile ja kataloge üle katta (overlay) läbipaistvalt.
- Failisüsteemi kasutav protsess näeb ühtset sidusat failisüsteemi

Docker konteinerite ketaste kihid



https://www.docker.com/blog/top-questions-containers-and-vms-together/

Docker konteinerite suurused

Platform	Distribution	Size (MB)
.NET Core	Alpine (runtime)	87
.NET Core	Alpine (runtime-deps)	54
.NET Core	Debian (runtime)	180
.NET Core	Debian (runtime-deps)	140
Node.js	Alpine	23
Node.js	Debian	202
OpenJDK 8 (JRE)	Alpine	57
OpenJDK 8 (JRE)	Debian (headless JVM)	79

Dockeri kasutamise eelised

- **Efektiivsus** - Vähem OS-i üldkulusid (overhead)
  - Ressursside jagamine
- **Mahutavatus** - Rohkem keskkondasid ühe serveri sisse (võrreldes VM'dega)
- **Teisaldavatus** - Vähem sõltuvusi platvormist = lihtsam liigutada infrastruktuuride vahel
- **Uuesti kasutatavus** – valmistame ette üks kord, kasutame mitu korda
- **Kiirus** – Ei ole vaja OS'i üles seada
- **Isoleeritavatus -** aga nõrgem kui VM puhul

Kokkuvõte

- VM'd ja konteinerid võimaldavad Infrastruktuur koodina (IaC)
- Virtualiseerimine
  - Pakub tugevamat isoleerimist
- Konteinerid
  - Ühtlane keskkond arendusest tootmiseni, liigutame konteinereid
  - Rakenduse isoleerimine ilma OS-i lisakoopiateta
  - Serverite konsolideerimine, kiired juurutused
  - Kiire juurutamine ja skaleerimine
- Pigem on vajadus kasutada Virtualiseerimist ning konteinereid käsi-käes

Selle nädala praktikum

- Raamatute API
  rakenduse konteineriseerimine

Järgmine loeng

- Hajus- ja Pilveandmebaasid