Виртуелизација и контејнеризација

Оперативни системи 2024

проф. д-р Димитар Трајанов,
проф. д-р Невена Ацковска,
проф. д-р Боро Јакимовски,
проф д-р Весна Димитрова,
проф. д-р Игор Мишковски,
проф. д-р Сашо Граматиков,
вонр. проф. д-р Милош Јовановиќ,
вонр. проф. д-р Ристе Стојанов,
доц. д-р Костадин Мишев

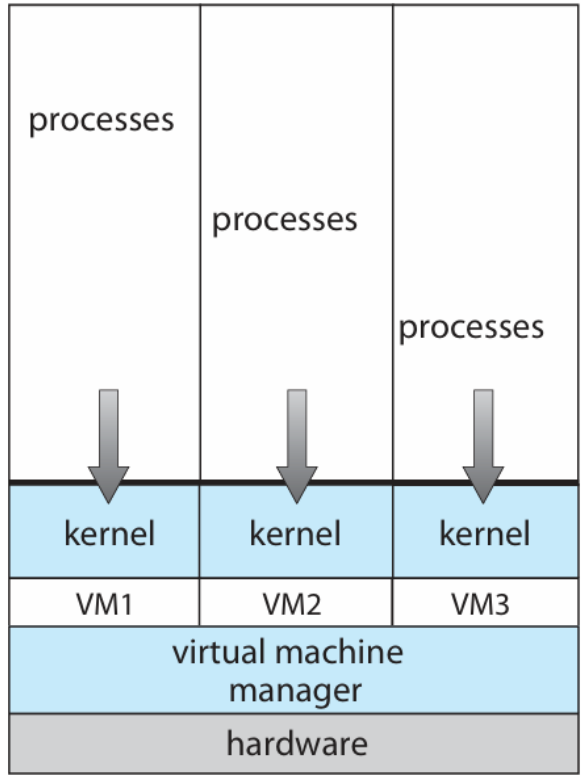
Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Вовед во виртуелизација

- Виртуелизацијата е техника која ги комбинира ресурсите од логичка перспектива за да овозможи нивна оптимална искористеност.
- Претставува процес за креирање на виртуелна верзија од нештото (пр. хардверот).
- Овозможува компјутерите да станат поискористени.
- Компјутерите кои користат виртуелизација ги оптимизираат своите достапни пресметковни ресурси.
- Виртуелизацијата овозможува еден компјутер да хостира повеќе виртуелни машини (ВМ) кои потенцијално може да имаат и различен оперативен систем (Linux, Windows...)
- Виртуелизацијата претставува ,,слој" на технологија кој се наоѓа помеѓу физичкиот хардвер на уредот и оперативниот систем, а со тоа да овозможи една или повеќе ,,виртуелни копии" на уредот.

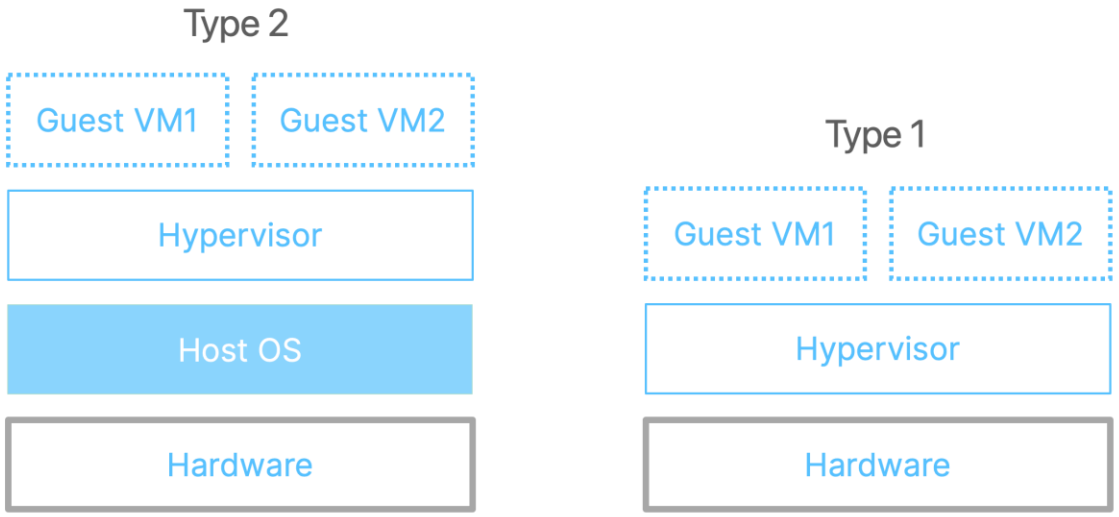
Виртуелна машина

- Виртуелната машина (ВМ) е софтверска имитација на хардверски систем која овозможува извршување на програми и оперативни систем.
- Виртуелизацијата креира виртуелен хардвер со клонирање на физичкиот хардвер.
- Секоја ВМ стартува изолирана околина на физичкиот хардвер и се однесува како комплетен компјутерски систем со свои сопствени компоненти.
- Во една ВМ, оперативниот систем со апликациите може да се извршуваат како на физички компјутер и апликациите во рамки на ВМ не ја забележуваат виртуелизацијата.
- Една виртуелна машина кореспондира на множество од датотеки.
- Еден **хипервизор** (virtual machine manager) овозможува симултана работа на повеќе различни оперативни системи во рамки на еден компјутер.
- Системските барања од ОС транспарентно се пречекуваат од софтверот за виртуелизација, и соодветно се предаваат на постоечкиот физички или емулиран хардвер.



Хипервизор

- Софтвер кој е инсталиран да работи над хардверот.
- Хостира виртуелни машини.
- Тип 1 хипервизор - хипервизор кој работи на ниво над хардверот.
  - VMware ESxi
- Тип 2 хипервизор - хипервизор кој работи над постоечки оперативен систем (хостиран хипервизор)
  - VMware Workstation



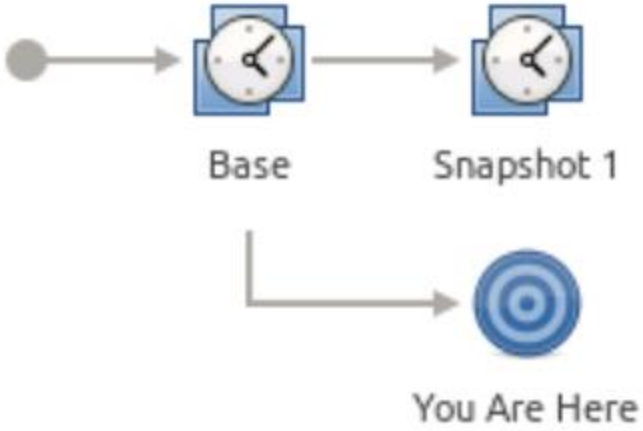
Датотеки поврзани со виртуелните машини

- ВМ може да се преместуваат низ различни хостови
- Датотеките се креираат од страна на хипервизорот и се зачувуваат во специјални именици.
- Примери за ВМ датотеки:

File Type	File Name	Description
Log File	<vmname>.log	Keeps a log of VM activity
Disk File	<vmname>.vmdk	Stores content of VM's disk drive
Snapshot Files	<vmname>.vmsd and <vmname>.vmsn	Stores information about VM snapshots (saved VM state)
Configuration File	<vmname>.vmx	Stores information about VM name, BIOS, guest OS, and memory

VM snapshot

- При работа со ВМ, понекогаш е потребно да се зачува прогресот или состојбата.
- Snapshot претставува множество на датотеки кои се зачувуваат во специјални именици со специјални екстензии (VMware чува .vmx)
- Што се чува во snapshot?
  - Состојбата на дисковите на виртуелната машина
  - Тековната содржината во RAM меморијата
  - Конфигурација и поставки

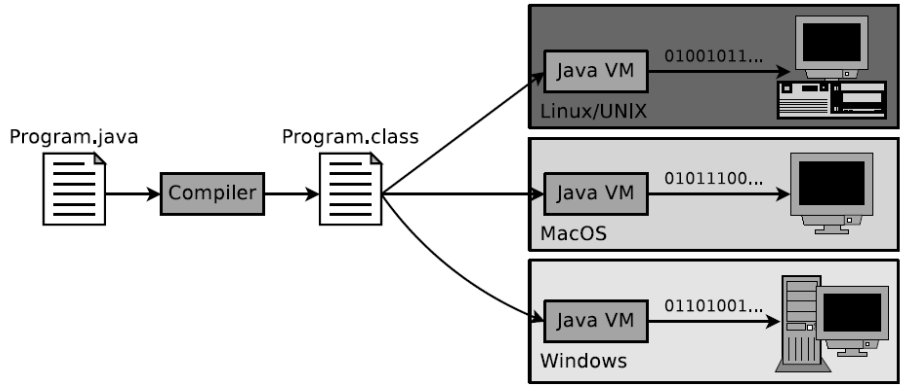


Типови на виртуелизација

- Виртуелизација на ниво на апликација (JavaVM)
- Партиционирање (делење на хардверските ресурси)
- Паравиртуелизација (Тип-1 хипервизор)
- Целосна виртуелизација (Тип-2 хипервизор)
- Виртуелизација подржана од харверот (Тип-0 хипервизор)
- Контејнеризација

Виртуелизација на ниво на апликација

- Виртуелизацијата на ниво на апликација овозможува секоја апликација да се стартува во виртуелна околина која ги овозможува сите компоненти кои апликацијата ги побарува.
- Виртуелната машина е поставена помеѓу апликацијата и оперативниот систем.
  - Java Virtual Machine (JVM).
- Предност: се овозможува независност од платформата.
- Недостаток: полоши перформанси во однос на нативно извршување на програмата.



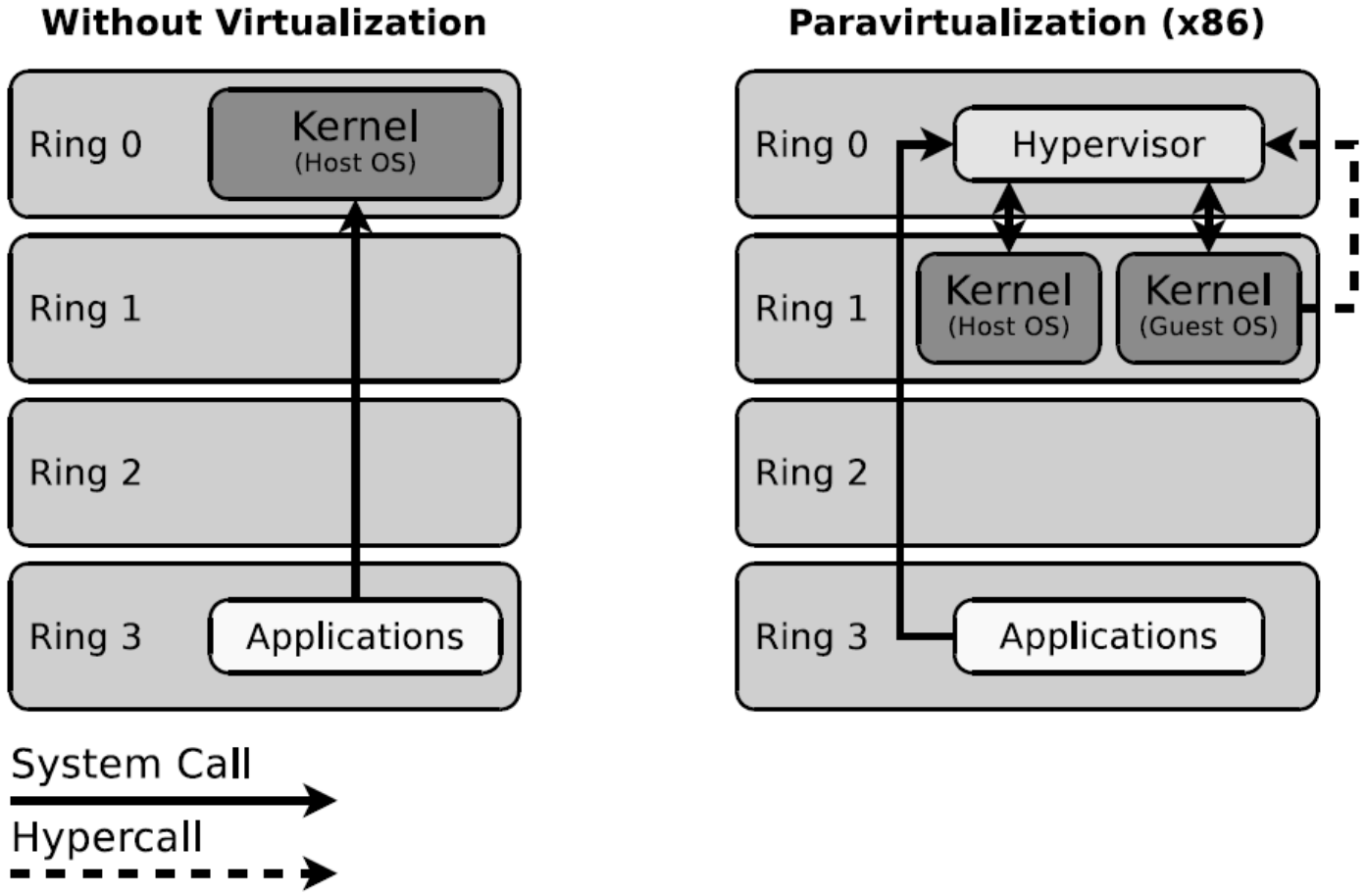
Партиционирање (делење на хардверот)

- Ако се користи партиционирање, хардверските ресурси може да се поделат за да се креираат подсистеми во еден компјутерски систем кој ќе си има сопствени ресурси.
- Секој систем може да содржи инстанца од ОС и може да се користи на истиот начин како независен компјутерски систем.
- Ресурсите (CPU, RAM, Disk...) се менаџираат од фирмверот на компјутерот кој ги алоцира за потребите на виртуелната машина. Не е потребен дополнителен софтвер за имплементација на овој тип на виртуелизација.

Паравиртуелизација (Тип-1 хипервизор)

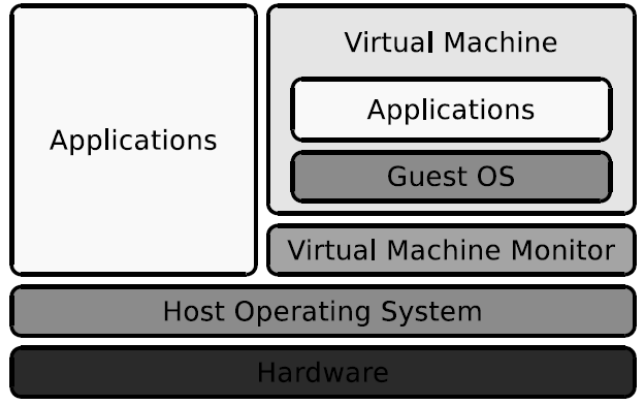
- При паравиртуелизација, ОС-гостин користи апстрактен слој за менаџирање, хипервизор, за пристап до физичките ресурси.
  - Тип-1 хипервизор
- Хипервизорот ги дистрибуира хардверските ресурси помеѓу гостинските ОС на ист начин како и оперативниот систем со стартуваните процеси.
- Хипервизорот работи во привилигираниот Прстен 0.
- Задолжително е потребен и ОС-домаќин кој ги поседува драјверите на уредите и работи на помалку привилигираниот Прстен 1.
- Поради тоа што јадрото на ОС-домаќин не може да извршува привилигирани инструкции поради неговата позиција во Прстен 1, хипервизорот прави т.н. хипер-повици.
  - Кога одредена апликација повикува системски повик, наместо системскиот повик, ОС-домаќин повикува заменска функција кај хипервизорот (некој вид на API за системски повици).
  - Хипервизорот нарачува извршување на системскиот повик на јадрото на ОС-домаќин.
  - Наместо оригиналните системски повици, ОС-гостин треба да повикува други повици од хипервизорот -> побарува модификација во јадрото на ОС-гостин (овозможено само за одредени ОС!)
- Примери: Xen, Citrix Xenserver, Virtual Iron, и VMware ESX Server.

Паравиртуелизација



Целосна виртуелизација (Тип-2 хипервизор)

- Софтверските решенија кои нудат комплетна виртуелизација овозможуваат секоја виртуелна машина да поседува комплетна виртуелна компјутерска околина, вклучувајќи и сопствен BIOS.
- Секој ОС-гостин има своја виртуелна машина со виртуелни ресурси (CPU, RAM, Storage, Network...).
- Основата на ова решение е посебна апликација наречена Виртуелен монитор на машината (ВММ) која е инсталирана во ОС-домаќинот.
  - Тип-2 хипервизор.
- Целта на ВММ е да алоцира хардверски ресурси за виртуелните машини
  - Некои хардверски компненти се емулирани (мрежните адаптери).
- Примери за ВММ:
  - Kernel-based Virtual Machine (KVM), Mac-on-Linux, Microsoft Virtual PC, Oracle VirtualBox, VMware Server, VMware Workstation, etc.



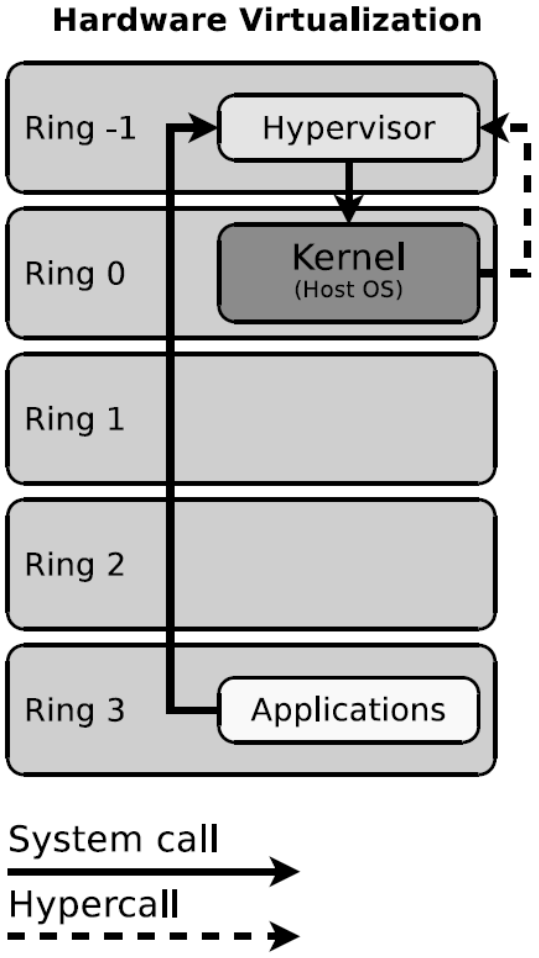
Целосна виртуелизација (Тип-2 хипервизор)

- ОС-гостин се извршува во user mode
- Како јадрото на ОС-гостин ќе извршува системски повици кога се наоѓа во user mode?
- Се користи бинарно преведување (binary tranlation) на системксите повици со инструкции кои можат да се извршуваат само во kernel mode
- Во текот на извршувањето, VMM ги скенира инструкциите кои следат после тековната инструкција. Ако детектира повик на сензитивни инструкции, ги заменува со функциски повици кои прават повици на ОС-гостин (кој работи во kernel mode) и има привилегии да ги изврши

Хардверска виртуелизација

- Модерните x86-компатибилни процесори (Intel и AMD) имплементираат екстензии за поддршка на хардверска виртуелизација.
  - Екстензијата резултира со модификација на нивоата со привилегии преку додавање на дополнителен Прстен -1 кој е наменет само за хипервизорот.
  - Освен user и kernel mode, додава guest и host mode (дозволува јадрото на guest оперативниот кој работи во user mode да извршува повици)
- Хипервизорот или ВММ кој работи на Прстен -1, постојано има контрола над CPU и на останатите хардверски ресурси, бидејќи Прстен -1 има повисоки привилегии во однос на Прстен 0.
- Виртуелните машини работат на Прстен 0 и се наречени Хардверски виртуелни машини (ХВМ).
- Предност во хардверската виртуелизација е тоа што изворниот код на ОС-гостин не треба да се менува.
  - Windows може да се стартува како ОС-гостин.
- Примери: Xen since version 3, Windows Server since version 2008 (Hyper-V), VirtualBox and KVM.

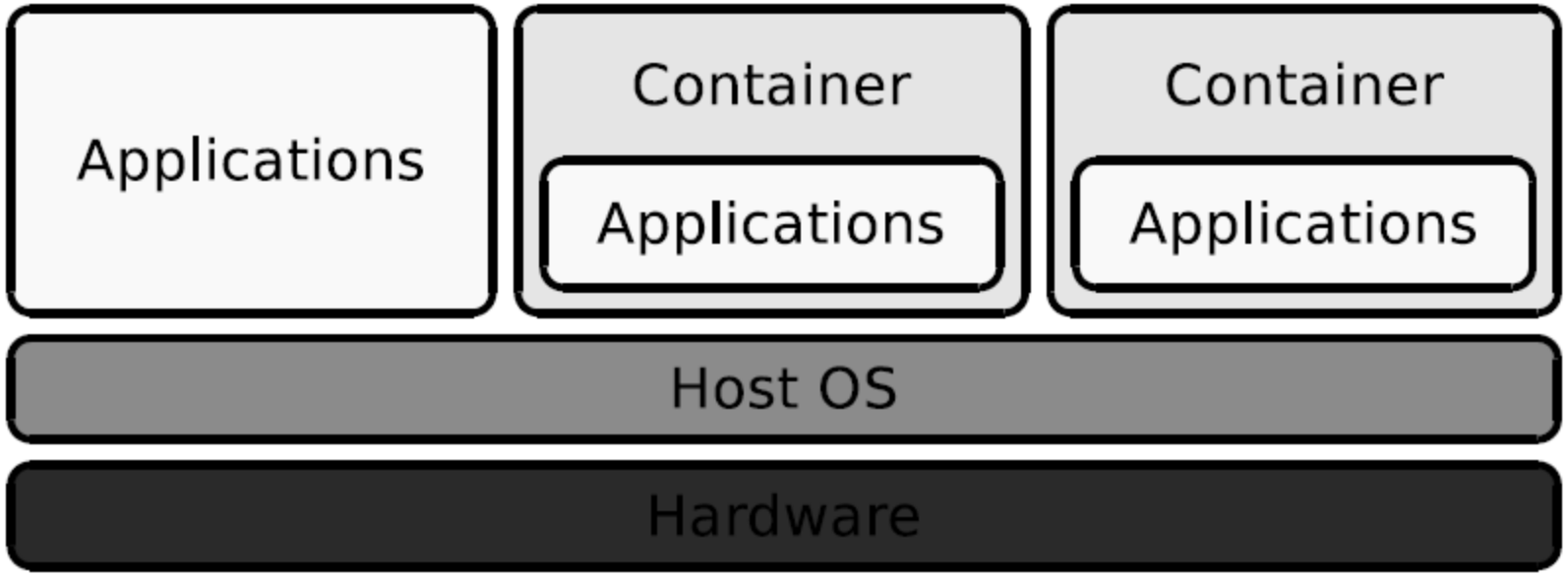
Хардверска виртуелизација



Контејнеризација

- Преку виртуелизација на ниво на ОС, повеќе идентични системски околини може да се стартуваат на исто јадро, изолирани еден од друг.
  - контејнери
- Кога се стартува ВМ, за разлика од целосната виртуелизација, паравиртуелизација или емулација, не се стартува нов ОС, туку се креира изолирана работна околина.
  - Сите конјтенери го користат истото јадро.
- Апликациите кои работат во контејнер може да соработуваат само со апликациите во истиот контејнер.
  - Предност: Намалено оптоварување бидејќи јадрото го менаџира хардверот.
- Овој концепт на виртуелизација е корисен во ситуации каде апликациите треба да работат во изолирани околини со висока безбедност.
  - Или, автоматизирана инсталација на апликациски софтвер, како веб сервер или база на податоци.
- **Docker**, Solaris, OpenVZ, FreeBSD, Virtuozzo...

Виртуелизација на ниво на оперативен систем - Контејнеризација



Контејнеризација

- Сегрегација на одговорности во DevOps
  - Развивачите на софтвер - се грижат за нивните апликации кои се извршуваат во контејнери;
  - Оператори - се грижат за работата на контејнерите;
- Ефикасен животен циклус на развој
  - Редуцирање на времето за пишување на код, тестирање и испорака;
  - Овозможува портабилност на апликациите;
- Поттикнува користење на сервисно-ориентирана и микросервисна архитектура
  - Поедноставна дистрибуција, скалирање и дебагирање;

Контејнеризација

Контејнери

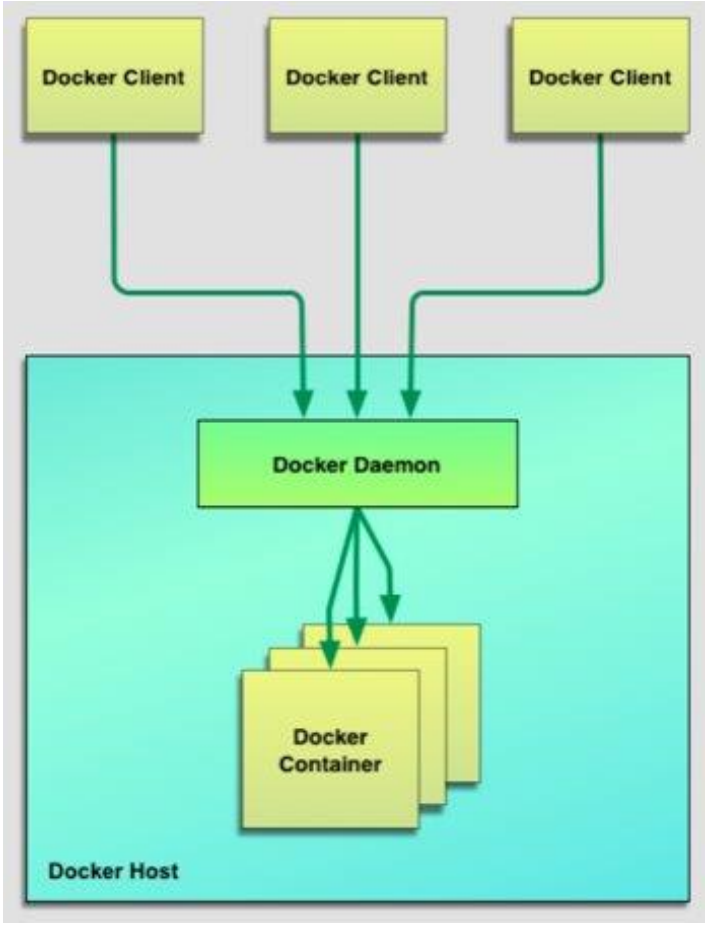
- Работи во корисничкиот простор на оперативниот систем --> Виртуелизација на ниво на ОС;
- Го користи интерфејсот за системски повици на ОС

Docker

- Софтвер со отворен код кој го автоматизира распоредувањето на апликациите во контејнери;
- Овозможува **строга изолација помеѓу контејнерите, нивната мрежа, склад на податоци и менаџмент на ресурси**;
- Додава механизми за интерација на врвот на виртуелизирана околина за извршување на контејнер;
- Едноставен, лесен и брз.
- **Не побарува емулатор или хипервизор слој** - го користи интерфејсот за системски повици овозможен од ОС-домаќин.
  - Ги намалува додатните побарувања потребни за стартување на контејнери и со тоа остава простор за стартување на поголем број на контејнери кои работат на домаќинот.

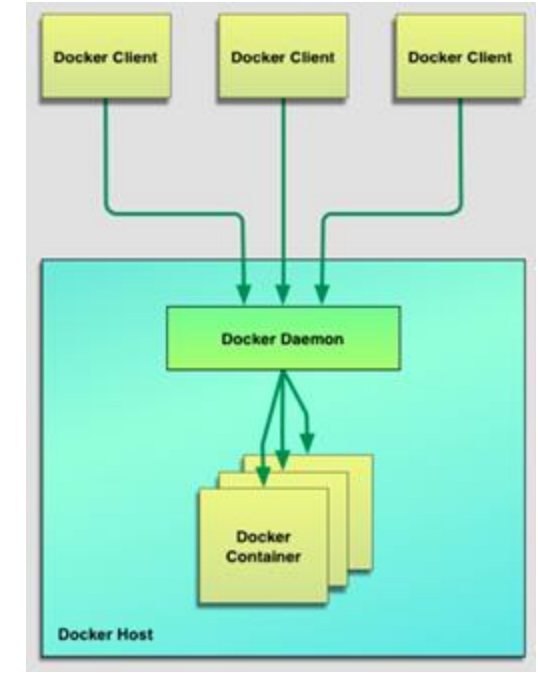
Архитектура на Docker (1)

- Docker компоненти:
  - Docker Engine (Docker client and server);
  - Docker Images;
  - Registries;
  - Docker Containers;
- Docker Engine
  - Docker клиентот зборува со Docker серверот (client-server app);
  - Docker серверот a.k.a. Docker Engine, или Docker daemon;
  - CLI binary: `docker`;

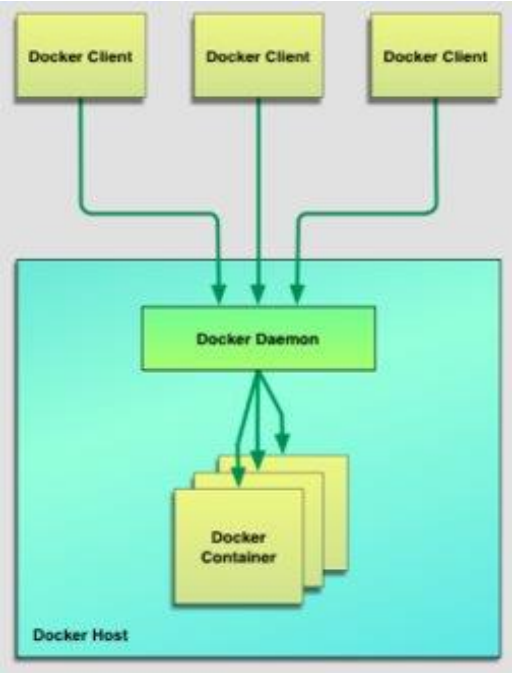


Архитектура на Docker (2)

- Docker Images
  - Градбени блокови во Docker светот;
  - Слоевит формат, градени чекор-по-чекор:
    - Додади датотека;
    - Изврши команда;
    - Отвори порта;
  - Претставуваат ,,изворен код" за Docker контејнерите;
- Регистри
  - Репозиториуми за чување на Docker Images;
  - Јавни и приватни;
  - Docker Hub: https://hub.docker.com;



Архитектура на Docker (3)



- Docker контејнери
  - контејнер= пакет за една апликација / сервис;
  - Контејнер се стартува од Docker Image;
    - Image = градбен блок;
    - Container = извршувачки блок;
  - Може да се креира, стартува, стопира, рестартира и уништува;
  - Docker не се грижи за што се извршува во контејнерот
  - Docker не се грижи каде ќе биде испорачан контејнерот;
- Docker мрежа
  - Docker мрежите се виртуелни мрежи кои се користат за да се поврзат повеќе контејнери.
  - Овозможуваат контејнерите да соработуваат помеѓу себе и со ОС-домаќин.

Docker Архитектура (4)

- Dockerfile
  - Dockerfile претставува скрипта со инструкции за како да се создаде Docker Image.
  - Содржи информации за оперативниот систем, јазиците, системските променливи, локациите на датотеките, мрежни порти и други детали кои се потребни за да се стартува сликата.
  - Командите во датотеката се поставуваат во групи, и овие групи се извршуваат автоматски.
- Docker Compose
  - Docker Compose е алатка која се користи за дефинирање и стартување на Docker апликации кои работат во повеќе контејнери.
  - Овозможува корисничко дефинирање на сервисите кои ги користи апликацијата во една единствена датотека.
  - Docker Compose овозможува едноставна конструкција, стартување и менаџирање на апликациите кои се дел од поголемо софтверско решение, без оглед на нивниот број.
  - Корисникот може да ги стартува одеднаш, да ги стопира, рестартира, гледа нивните логови, менаџира нивната приватна мрежа итн.
- Docker Swarm
  - Docker Swarm е сервис кој овозможува оркестрација помеѓу контејнерите, нивна распределба и извршување низ кластер на сервери.
  - Kubernetes - позната алтернатива на Docker Swarm

Docker Ecosystem (1)

- Docker (Docker Engine)
  - Основнтата технологија позади Docker;
  - Софтвер со отворен код
  - Овозможува стартување на контејнери врз Linux јадро;
- Docker CLI
  - CLI овозможува интеракција помеѓу развивачите на софтвер и Docker Engine, преку `docker` и `docker-compose` командите;
  - open-source;

The Docker Ecosystem (2)

- Docker Desktop
  - Софтвер со затворен код, кој овозможува целосна околина за работа со Docker технологиите: контејнери, слики, итн;
  - Наменето за Windows и MacOS корисниците;
- Docker Hub
  - Јавен репозиториум за зачувување, пребарување и споделување на Docker Images;
- Docker, Inc.
  - Компанија во САД, која е сопственик на Docker Desktop и Docker Hub;

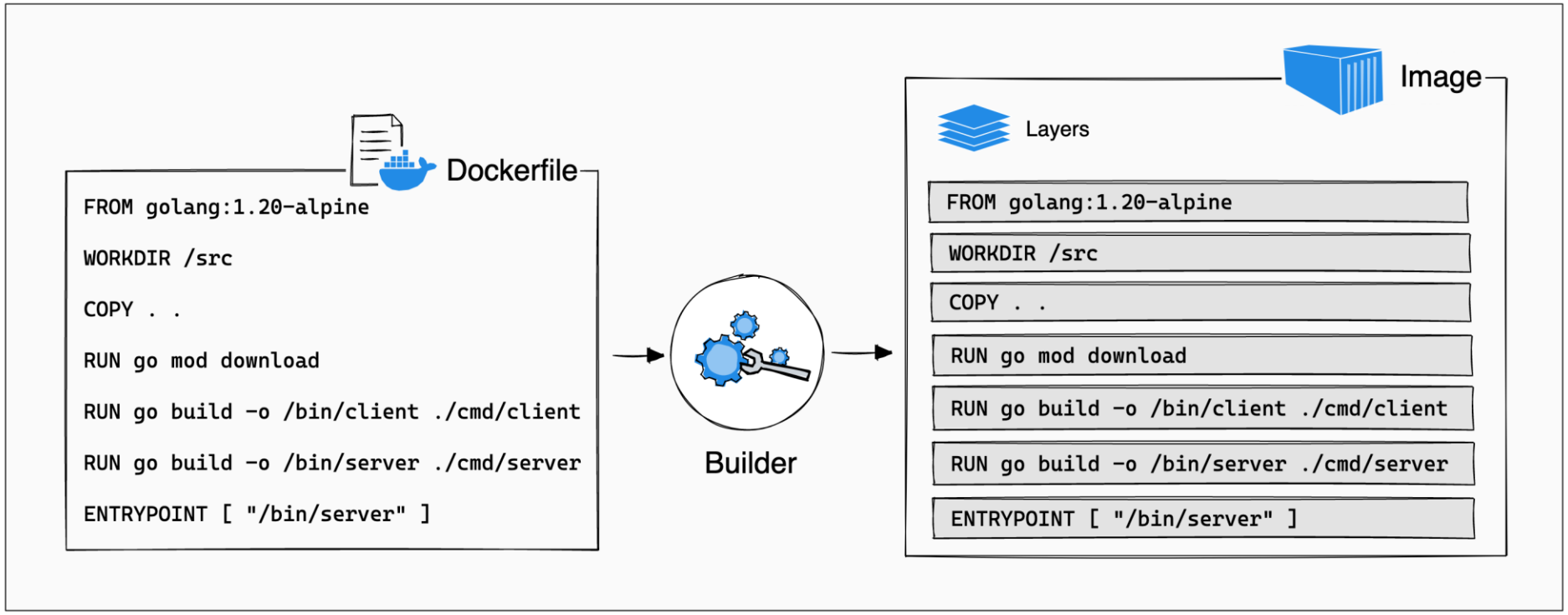
Docker Image

- Docker image претставува read-only шаблон кој содржи множество на инструкции за креирање на контејнер кој може да се стартува на Docker платформата.
- Обезбедува пригоден начин за пакување апликации и претходно конфигурирани серверски околини, кои можете да ги користите за ваша приватна употреба или јавно да ги споделувате со други корисници на Docker.
- Составен од колекција на датотеки, како што се **инсталации**, **код на апликација** и **зависности**.
- Docker Image може да се креира со еден од следните два методи:
  - Интерактивно - Преку стартување на контејнер од постоечки Docker image, рачно менување на околината на контејнерот преку извршување на серија од операции, и зачувување на резултантната состојба како нова слика.
  - Dockerfile - Преку составување на текстуална датотека, позната како Dockerfile, која претставува спецификација за креирање на Docker контејнер.

Docker слоеви

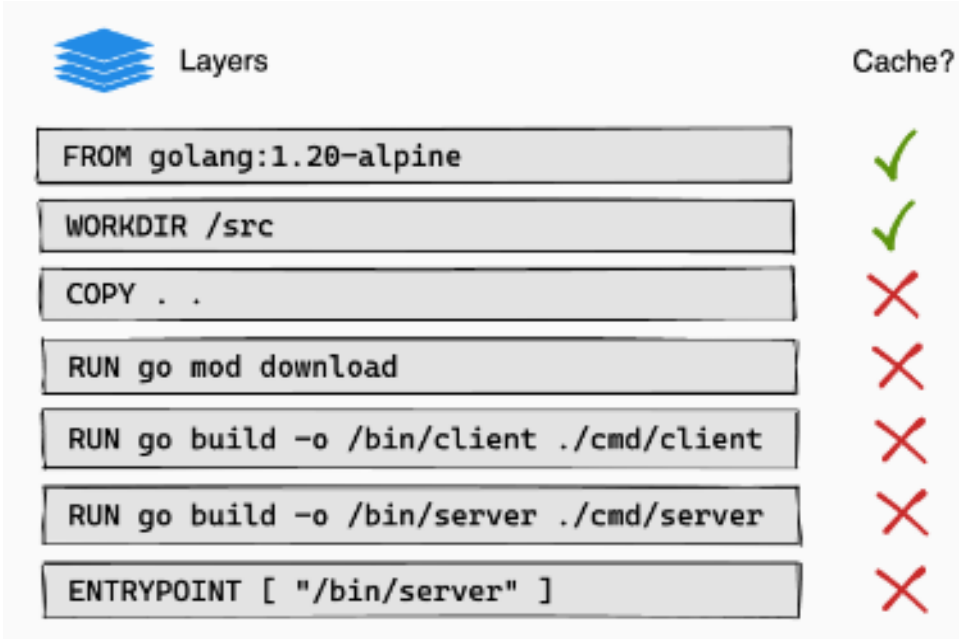
- Секоја од датотеките што ја сочинуваат Docker image е позната како слој. ,
- Овие слоеви формираат серија на меѓу-слики, изградени една врз друга во фази, каде што секој слој зависи од слојот веднаш под него.
- Хиерархијата на слоевите е клучна за ефикасно управување со животниот циклус на Docker image.
  - Слоевите кои најчесто се менуваат теба да бидат колку што е можно повисоко во стекот. На овој начин ќе се заштеди време при rebuild.
- Слој на контејнерот
  - При стартување на контејнер од слика, Docker додава тенок слој на кој може да се менува, познат како слој на контејнер, кој ги зачувува сите промени во контејнерот во текот на неговото извршување.
- Родителска слика (Parent Image)
  - Првиот слој на Docker image е познат како ,,parent image”.

Docker слоеви

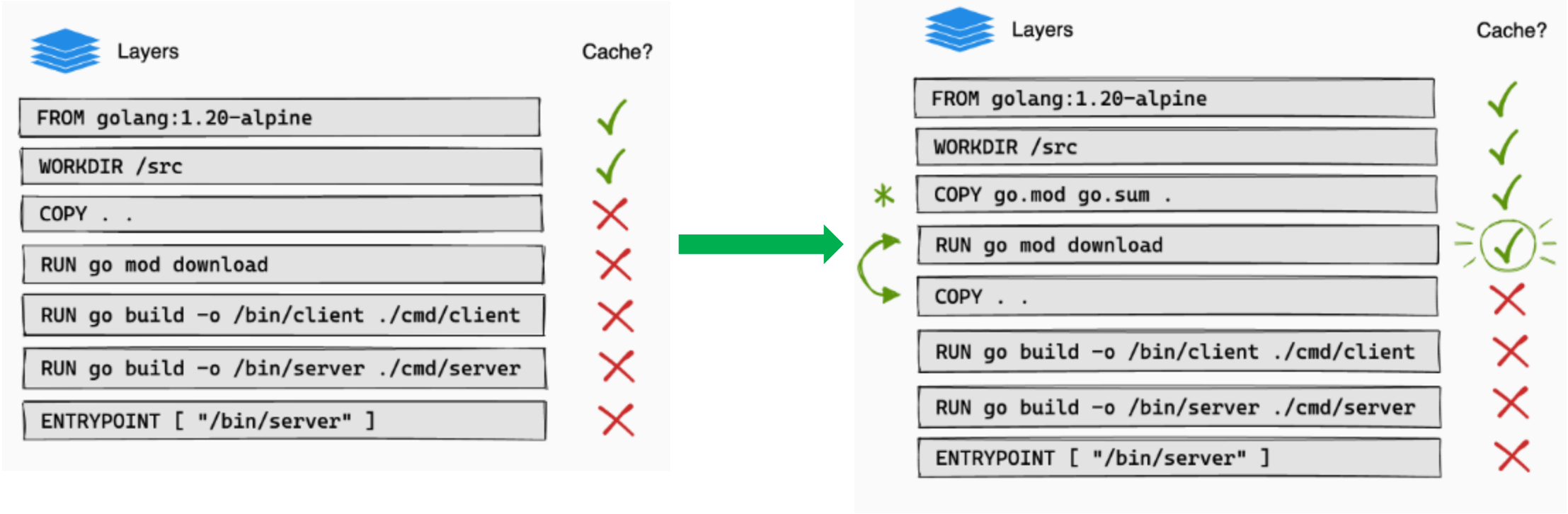


Кеширани слоеви

- При build-ање на Docker image од Dockerfile, Docker се обидува да ги реискористи слоевите од претходните извршувања.
- Ако слојот на сликата не е променет, docker ја искористува старата кеширана слика.
- Ако слојот се променил од претходното build-ање, docker одново ќе го изгради тој слој.



Кеширани слоеви - Пример за оптимизација



Docker команди – Менаџмент на контејнери

`docker ps`	List the running containers
`docker ps -a`	List all the containers, both running and stopped.
`docker create [image]`	Create a container without starting it.
`docker create –it [image]`	Create an interactive container with pseudo-TTY
`docker rename [container] [new-name]`	Rename a container
`docker rm [container]`	Remove a stopped container.
`docker rm –f [container]`	Force remove a container, even if it is running
`docker logs [container]`	View logs for a running container

Docker команди – Стартување на контејнер

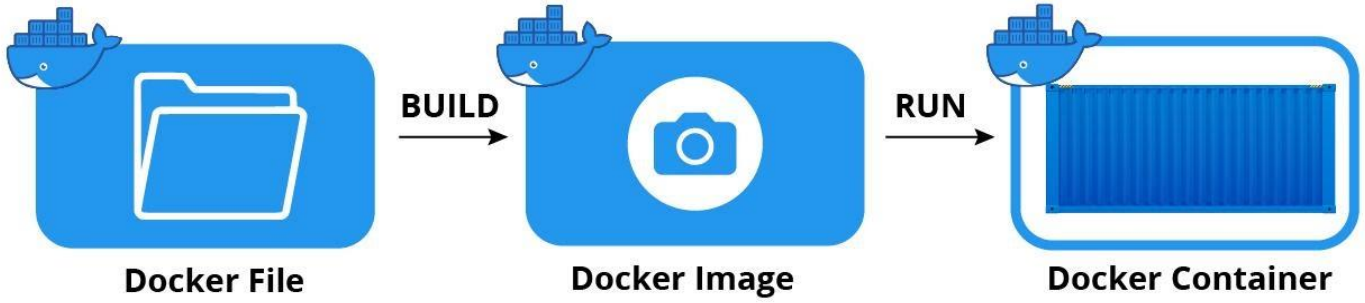
`docker run [image] [command]`	Run a command in a container based on an image.
`docker run –name [container-name] [image]`	Create, start and name a container.
`docker run –p [host]:[container-port] [image]`	Map a host port to a container port.
`docker run –rm [image]`	Run a container and remove it after it stops.
`docker run –d [image]`	Run a detached (background) container.
`docker run –it [image]`	Run an interactive process, e.g. a shell in a container.
`docker start [container]`	Start a container.
`docker stop [container]`	Stop a container
`Docker restart [container]`	Restart a container.
`docker exec –it [container] [shell]`	Run a shell inside a running container.

Docker команди– Менаџмент на Docker Images

`docker build [dockerfile-path]`	Create an image from a Dockerfile
`docker build .`	Build an image using the files from the current path.
`docker build –t [name]:[tag] [location]`	Create an image from a Dockerfile and tag it.
`docker build -f [file]`	Specify a file to build from.
`docker pull [image]`	Pull an image from registry.
`docker push [image]`	Push an image to a registry.
`docker commit [container] [new-image]`	Create an image from a container.
`docker images`	Show all locally stored top level images.
`docker rmi [image]`	Remove an image.
`docker images prune`	Remove unused images.

Методи за градење на Docker image

- Интерактивно
  - Предности: Најбрзиот и наједноставниот начин за креирање на Docker слики. Идеален за тестирање, решавање проблеми, одредување зависности и валидација на процеси.
  - Недостатоци: Тешко управување со животниот циклус, побарува постојана рачна реконфигурација.
- Dockerfile
  - Овозможува градење на чисти, компактни и повторливи слики базирани на точно дефинирани рецепти.
  - Полесно управување со животниот циклус и полесна интеграција во процесите на континуирана интеграција (CI) и континуирана испорака (CD).



Интерактивен метод - Пример

- Стартувај интерактивна shell сесија на контејнер. Контејнерот се заснова на Docker image со Debian OS преземен од Docker Hub:
  - $ docker run -it debian:11-slim bash
- Инсталирај nginx сервер:
  - $ apt-get update && apt-get install -y nginx
- Излези од интерактивен режим:
  - Ctrl + D
- Излистај ги сите docker контејнери (вклучувајќи ги и стопираните):
  - $ docker ps –a

  output:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
61d961dfdf7b	debian:11-slim	"bash"	About a minute ago	Exited (0) 6 seconds ago		

- Зачувај ја состојбата на контејнерот во слика со docker commit наредбата:
  - $ docker commit 61d961dfdf7b debian_nginx

- Излистај ги docker images за проверка дали операцијата е успешна:
  - $ docker images

  output:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
debian_nginx	latest	538a64303a41	8 seconds ago	164MB

Dockerfile команди

Команда	**Опис**
FROM	To specify the parent (base) image.
WORKDIR	To set the working directory for any commands that follow in the Dockerfile.
RUN	To install any applications and packages required for your container.
COPY	To copy files or directories from a specific location on the host to a location in the container
ADD	As COPY, but also able to handle remote URLs and unpack compressed files.
ENTRYPOINT	Command that will always be executed when the container starts. If not specified, the default is /bin/sh -c
CMD	Arguments passed to the entrypoint. If ENTRYPOINT is not set (defaults to /bin/sh -c), the CMD will be the commands the container executes.
EXPOSE	To define which port through which to access your container application.
LABEL	To add metadata to the image.

Dockerfile - Пример

Содржина на Dockerfile датотеката

```
# Use the official debian:11-slim as base
FROM debian:11-slim
# Install nginx and curl
RUN apt-get update && apt-get upgrade -y && apt-get install -y nginx curl
&& rm -rf /var/lib/apt/lists/*
```

Користи ја docker build командата за креирање на слика од Dockerfile. Со -t знаменцето се сетира име на сликата и таг:

```
$ docker build -t my-nginx:0.1 .
```

Изврши ја docker images командата за преглед на ново креираната слика:

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-nginx	0.1	f95ae2e1344b	10 seconds ago	164MB

Заклучок (Виртуелизација) ...

- Подобрено искористување на ресурсите: Овозможува поефикасно користење на хардверските ресурси дозволувајќи им на повеќе виртуелни машини да работат на еден физички сервер.
- Изолација и безбедност: Обезбедува силна изолација помеѓу виртуелните машини, ја подобрува безбедноста преку спречување на ширење на ранливости и обезбедувајќи подобра заштита од потенцијалните закани.
- Флексибилност и приспособливост: Овозможува лесно скалирање на ресурсите, обезбедува флексибилност во прилагодувањето на променливите оптоварувања.
- Заштеда на трошоци: со консолидирање на повеќе оптоварувања на еден сервер, трошоците за хардвер, потрошувачката на енергија и целокупните трошоци за инфраструктура се намалуваат.
- Обнова и резервна копија од катастрофи: Олеснува ефикасни стратегии за обновување при катастрофи со овозможување создавање снимки и резервни копии на виртуелни машини.
- Подобрено одржување и надградби: Ги поедноставува задачите за одржување и надградбите на системот со тоа што овозможува лесна миграција на ВМ помеѓу физичките сервери, минимизирајќи ги времето на прекин и прекини.

... Заклучок (Виртуелизација)

- Поддршка за компатибилност: Овозможува коегзистенција на различни оперативни системи и наследни апликации на ист физички хардвер.
- Поделба на ресурси: Овозможува поделба на ресурсите, како што се процесорот, меморијата и складирањето, обезбедувајќи правична и ефикасна распределба меѓу виртуелните машини за оптимални перформанси.
- Интеграција во облак: Служи како основна технологија за пресметување во облак, овозможувајќи го создавањето и управувањето со виртуелни инстанци во облак за зголемена приспособливост и флексибилност.
- Подобрен развој и тестирање: Обезбедува контролирана и повторлива средина за развој и тестирање на софтвер, без потреба од дополнителен физички хардвер.

Заклучок (Docker)

- Docker е моќна алатка за креирање, распоредување и управување со контејнеризирани апликации во широк опсег на средини.
- Ја подобрува конзистентноста, преносливоста и приспособливоста во процесите на развој и поставување на софтвер.
- Обезбедува флексибилен и ефикасен начин за пакување на апликациите и нивните зависности.
  - Лесно преместување на кодот помеѓу средини за развој, тестирање и продукција;
- Голема и активна заедница на корисници и соработници.
- Многу важна алатка и за инженерите на DevOps и за развивачите на софтвер.

Прашања?