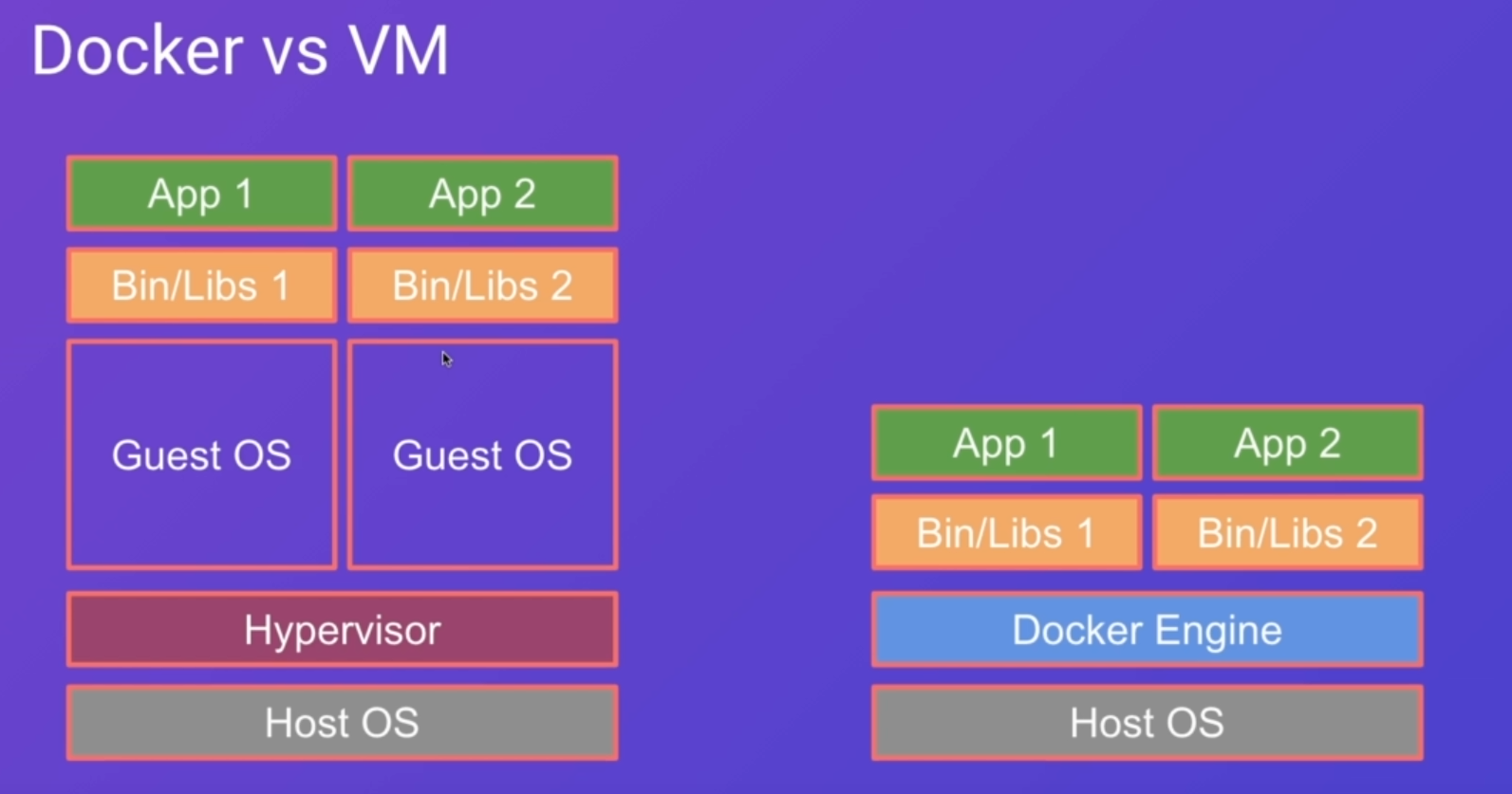
***DOCKER***



Сравнение с виртуальной машиной.

У нас есть хостовоя ОС на которой установлен Docker Engine или Hypervisor (гипервизор тоже может быть хостовой ОС).

Далее на Hypervisor-е запускаются гостевые ОС, которые внутри уже содержат бинарники и библиотеки необходимые для запуска приложения и само приложение. Если необходимо запустить два приложения, запускается дополнительная гостевая ОС, на которую уже размещают бинарники и библиотеки второго приложения и само второе приложение.

У Docker-а на хостовую ОС устанавливается Docker Engine (один) и на него устанавливаются библиотеки и приложения, при этом нет прослойки в виде гостевой ОС.

Docker не имеет ничего общего с виртуальной машиной (если только изолированность).

рассмотрим как работают процессы в линукс

есть процесс, в котором мы сейчас находимся и мы хотим запустить процесс на node(напр)что будет происходить, когда мы запустим процесс на node:

1. создастся новый процесс при этом скопировав предыдущий(дешёвая операция)был баш мы делаем форк и появляется второй баш,
2. далее как только до него доходит очередь он запускается и подменяется необходимым процессом. это делается с помощью execv.
3. у нового процесса появится procesID и он будет функционировать в рамках этого процесса.

В данном случае первый процесс не изолирован, он может получать доступ к диску, может что-то удалить, а если второй процесс будет использовать ту же область диска, то первый может удалить нужную второму информацию.

В линуксе предвесником изоляции была команда chroot

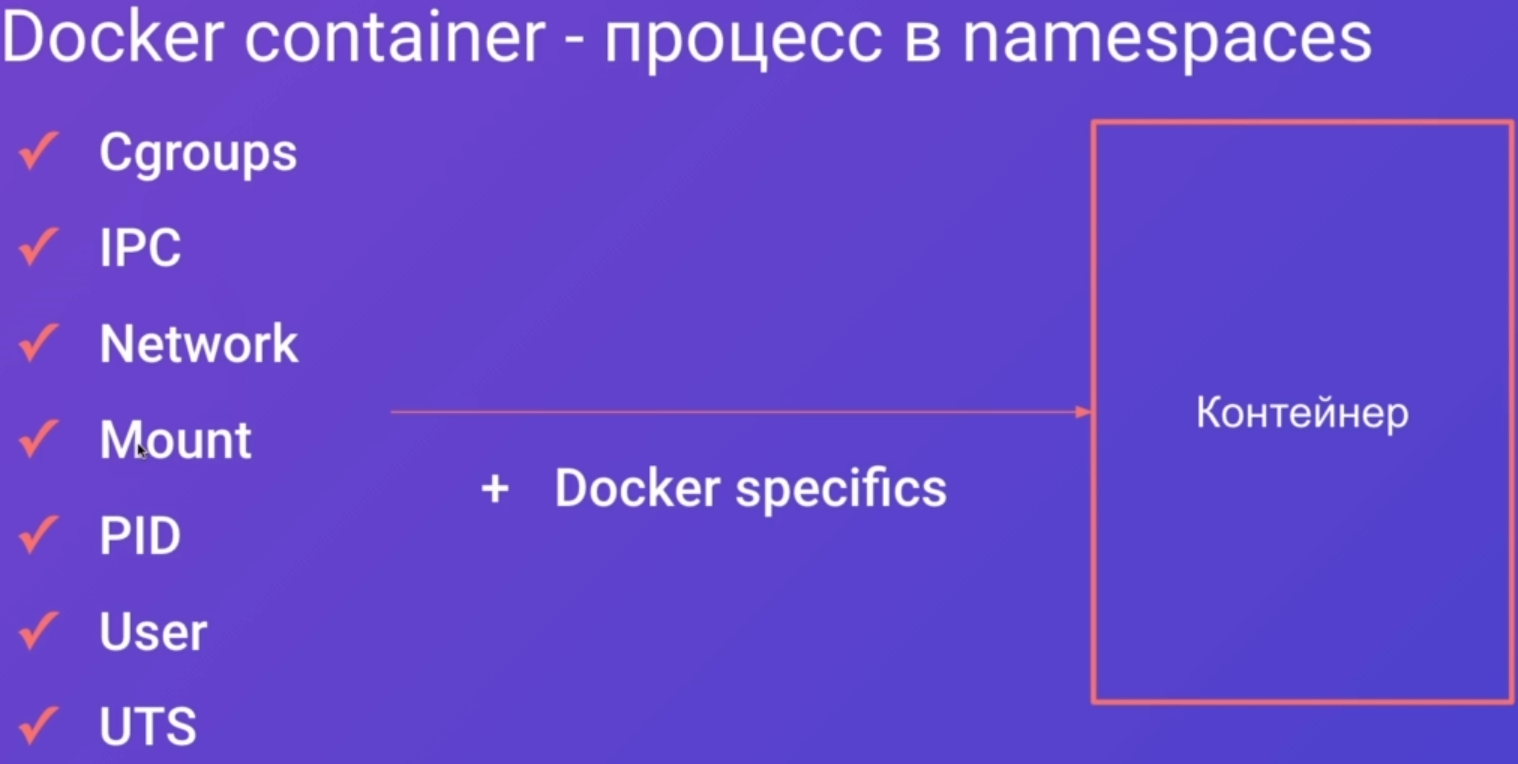
позволяет сменить корневую директорию

docker для изоляции использует namespace.

Но по аналогии можно сказать, что мы запустим процесс, а затем сменим ему root-овую директорию на к-л кастомную и он не будет видеть другие бинарники и библиотеки внутри своего изолированного пространства. В рамках этой изоляции мы можем скопировать другие бинарники и запускаться с ними в рамках этой изоляции не получая доступ к дисковому пространству.

что есть namespace?

когда запускается контейнер, создается новый namespace. namespace имеет несколько наборов это



* Cgroups управляет ограничениями по памяти и процессору для конкретного namespace.
* IPC – коммуникация между процессами
* Свой Network
* Mount – говорит какие директории доступны какие нет.
* processID который может повторяться с processID текущего Если мы запускаем на хосте processID должен быть уникален, если мы запускаем в namespace, то processID может быть свой
* User может быть аналогичный и он никак не будет конфликтовать с корневым user host машины
* и UTS

Кроме всего этого есть специфичные для docker обвязки

9:45

Контейнер это некая изолированная часть изолированный namespace, который запускается на ядре хостовой машины и фунциклирует изолированно, за исключением того, что он получает доступ к ресурсам машины

Благодаря изоляции можно запускать разные библиотеки и это не будет никак конфликтовать из-за изолированности контейтеров.

Контейнер это изолированный namespace с дополнительными обвязками docker, в котором запускаются приложения, кладутся библиотеки для работы с ядром нашей ОС, после запуска этот контейнер работает изолированно. При этом docker дает удобный api для взаимодействия.

ОГРАНИЧЕНИЕ: МЫ НЕ МОЖЕМ ЗАПУСКАТЬ ARM-ОВСКУЮ ОС НА x86 И ОБРАТНО. Поэтому если контейнер сбилджен для ARM, то и запускать его нужно на ARM-е.

Из чего состоит docker?

Состоит из 3 частей:

1 - клиент CLI

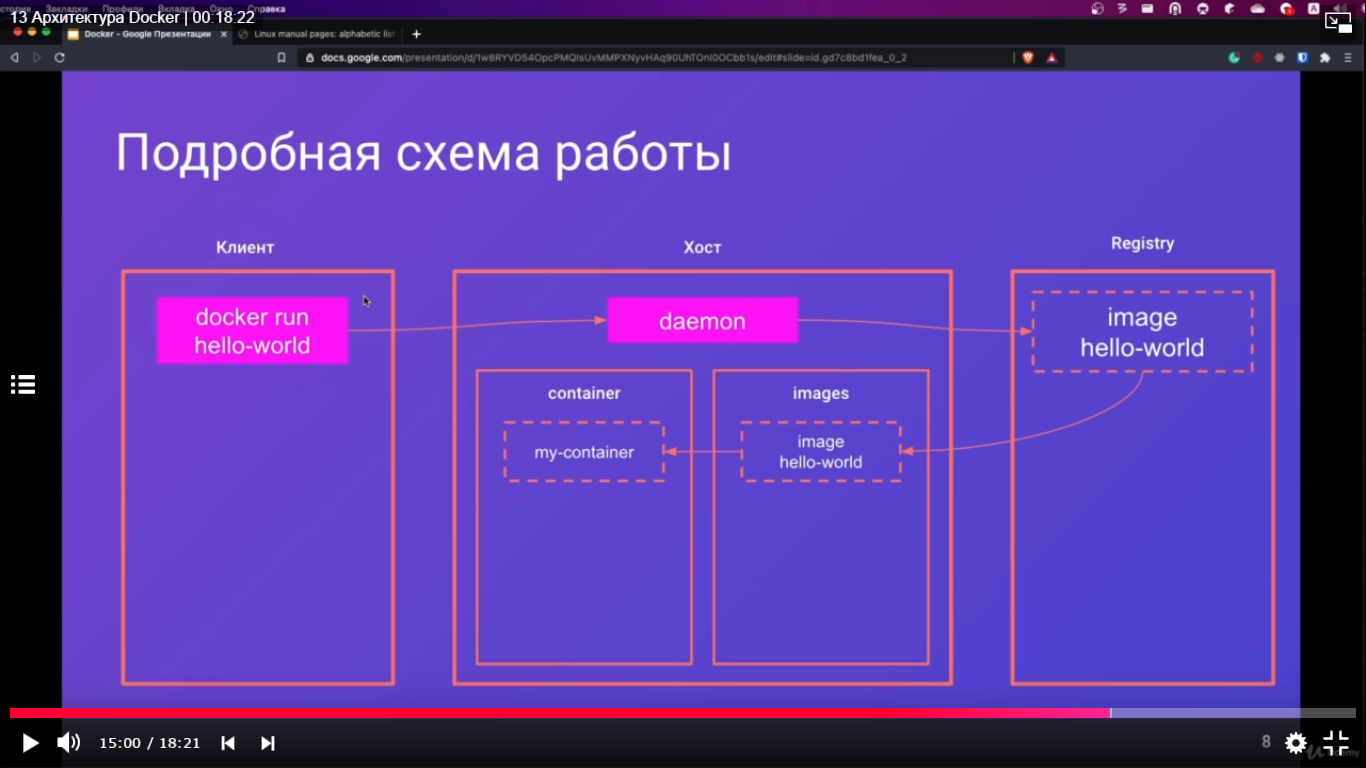
2 - API

3 - Docker host

Когда мы нажимаем docker ps мы на самом деле запускаем не docker, а пользуемся клиентом к docker. Это удобная CLI утилита, которая позваляет взаимодействовать с API docker-a.

Docker host это процесс хостовый (docker demon) который и работает на этой машине поверх него есть API и когда мы пишем docker ps, мы отправляем команду из нашей CLI на api и docker её обрабатывает. Можно даже написать свой CLI для взаимодействия с docker.

Схема работы



У нас есть клиент в рамках которого мы делаем docker run hello-world

После этого он делает запрос по api к хосту, на этом хосте крутится докер демон, он проверяет есть ли е нас такой имадж локально, если его нет идет в общий registry(интернет, также мы можем сами создавать регистри) после этого он скачивает нужный ему имадж и после этого он начинает запускать контейнер (создается новый namespace и передает туда этот имадж, он распаковывается ( бинарники, либы) всё что нужно чтобы этот имадж запустить и уже в этом запускается наше приложение