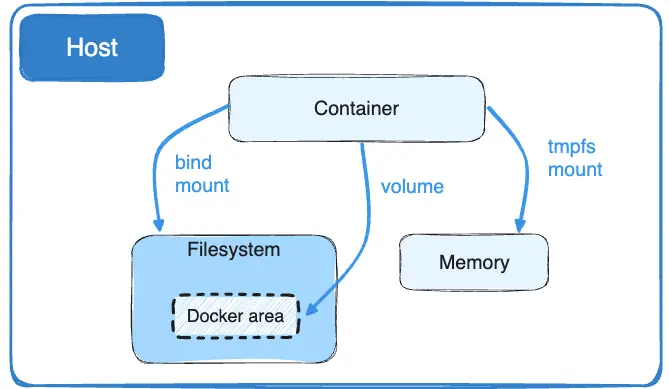
**+++++THEORY+++++**

https://docs.docker.com/engine/storage/

**Ephemeral File System** (inside container)

https://docs.docker.com/engine/storage/

Любой образ базируется на ОС, например Ubuntu, Debian, Alpine Linux, и наследует её файловую систему, которую можно использовать для хранения данных без дополнительного монтирования хранилища.

Данные доступны только из контейнера, удаляются вместе с контейнером.

!Внимание, у этого метода самая низкая скорость записи, смотри Doc по ссылке выше, и

нежелательный эфект в виде увеличения размера слоя для записи контейнера.

Используется для создания нового образа: к базовому образу добавляется слой для записи с

необходимыми файлами.

**Volumes** (тома)

https://docs.docker.com/engine/storage/volumes/

Изолированные хранилища данных управляемые исключительно docker которые могут быть смонтировать внутрь любого числа контейнеров но данные в volumes хранятся независимо от контейнера, физически находятся в /var/lib/docker/volumes.

Можно использовать для передачи данных между контейнерами, базы данных СУБД которой

развернуто в контейнере и других нужд stateful app.

Поведение: 1. анонимные тома удаляются автоматически. 2. не удаляются автоматически: именованные тома, именованные тома созданные при создании контейнера, анонимные тома

контейнеры которых не удаляются автоматически после завершения работы.

**Bind mounts**

Каталог или файл localhost смонтированный внутри контейнера.

Можно использовать при разработке: Laravel app находится в папке контейнера и доступно через папку localhost для редактирования в IDE.

Поведение: 1. при монтировании содержимое папки localhost заменяет содержимое папки контейнера; 2. изменение содержимого папки контейнера или localhost сразу отображается везде.

!Строго не рекомендуется для production: нарушение принципа изоляции, уязвимость данных.

**tmpfs mounts**

Аналогичен Volumes, но:

!Физически данные находятся в ОЗУ, возможна утечка памяти. Максимальный размер тома по умолчанию 50% от общего объема оперативной памяти localhost.

Используется только на localhost под linux. Доступ только из одного контейнера, быстрая скорость, удаляется при остановке контейнера.

Можно использовать для кеширования данных.

**Монтирование**

Происходит при выполнении команды docker run добавлением флага -v или его алиаса

--volume, --tmpfs или --mount с опциями.

**Формат записи флагов**

**-v** и **--volume**

Формат сокращенный до трех полей и поэтому имеет ограниченный функционал по сравнению с --mount, флаги разделяются **:** и должны быть в правильном порядке:

1. имя тома(volumes) или путь к папке(bind mounts); для анонимных томов поле должно отсуцтвовать вместе с разделителем(**:**).

2. путь, по которому файл или каталог смонтирован в контейнере.

3. необязательный флаг определяющий поведение монтируюмых хранилищ:

rw - чтения и записи (поведение по умолчанию, если флаг пропущен);

ro - только чтения;

z,Z - при использовании selinux, смотри по ссылке:

https://docs.docker.com/engine/storage/bind-mounts/#configure-the-selinux-label

**----tmpfs**

Формат сокращенный до двух полей и поэтому имеет ограниченный функционал по сравнению с --mount, флаги разделяются **:** и должны быть в правильном порядке:

1. путь монтирования в контейнере.

2. необязательный флаг с опциями:

size=1G - размер в байтах

mode=0755 - режим доступа к файла, по умолчанию 777

**--mount**

Принимает неограниченное количество флагов в формате key1=value1,key2=value2,… в любом порядке, имеет более многословный но более читаемый синтаксис.

**-----THEORY-----**

**docker volume create volumeName**

создать том с именем volumeName

**docker volume ls**

список томов

**docker volume inspect volumeName**

подробная информации по одному или нескольким томам

**docker volume prune**

удаление всех анонимных томов неиспользуемых ни одним контейнером

**docker volume rm volumeName**

удалить один или несколько томов с именем volumeName

**docker run -v volumeName:/path/to/container/directory:rw imageName**

**docker run --mount type=volume,src=volName,dst=/path/to/container/directory,ro imageName**

монтирование тома при помощи разных флагов

**-v** и **--volume keys:**

**/path/to/localhost/directory:/path/to/container/directory:rw**

смонтировать папку из localhost в контейнер с правами чтения и записи

**volumeName:/path/to/container/directory:ro**

смонтировать именованный том в контейнер с правами только чтение; если том с таким

именем отсутствует, то docker создаст его; такой том не удаляется после удаления контейнера

созданного с флагом docker run --rm

**/path/to/container/directory**

смонтировать анонимный том в контейнер с правами чтения и записи (по умолчанию),

анонимный том удаляется после удаления контейнера созданного с флагом docker run --rm

/path/to/localhost/directory:/path/to/container/directory:rw,**rprivate**

рекурсивное монтирование поддиректорий в монтируемой директории является поведением

по умолчанию, но может быть изменено, смотри доку:

https://docs.docker.com/engine/storage/bind-mounts/#configure-bind-propagation

https://docs.docker.com/engine/storage/bind-mounts/#recursive-mounts

**--tmpfs keys:**

**/path/to/container/directory:size=1G,mode=0755**

смонтировать tmpfs по пути в контейнер, размером 1 Гига байт (размер указывается в байтах),

режим доступа к файла 0755 (по умолчанию 777)

**--mount keys:**

**type=volume**

**type=bind**

**type=tmpfs**

тип монтируемого хранилища

**source=volumeName**

**src=volumeName**

смонтировать именованный том

**volume-subpath=/path/to/volume/directory**

смонтировать отдельную директорию смонтированного тома в контейнер

https://docs.docker.com/engine/storage/volumes/#mount-a-volume-subdirectory

**bind-propagation=rprivate**

рекурсивное монтирование поддиректорий в монтируемой директории является поведением

по умолчанию, но может быть изменено, смотри доку:

https://docs.docker.com/engine/storage/bind-mounts/#configure-bind-propagation

https://docs.docker.com/engine/storage/bind-mounts/#recursive-mounts

**source=/path/to/localhost/directory**

**src=/path/to/localhost/file.txt**

смонтировать папку или файл с localhost

**destination=/path/to/container/directory**

**dst=/path/to/container/directory**

**target=/path/to/container/directory**

путь в контейнере для монтирования хранилища

**readonly**

**ro**

режим хранилища «только для чтения»; ключ без параметров

**volume driver**

docker может использовать хранилища расположенные удаленно используя разные протоколы передачи данных: ssh, nfs server, samba, Block storage devices(USB и сетевые диски).

Тема обширная, смотри Doc и пример с volume-opt ниже

https://docs.docker.com/engine/storage/volumes/#use-a-volume-driver

**volume-opt=**key1=value1,key2=value2…

«дополнительные» опции тома; примеры:

volume-opt=device=<nfs-server>:<nfs-path>

"volume-opt=o=addr=<nfs-address>,vers=4,soft,timeo=180,bg,tcp,rw"

Для более подробного описания смотри Doc:

https://docs.docker.com/engine/storage/volumes/#choose-the--v-or---mount-flag

**tmpfs-size=32M**

размер tmpfs в байтах (по умолчанию до 50% RAM localhost)

**tmpfs-mode=1770**

режим доступа к файлам (по умолчанию 777)

**docker run --volumes-from containerName imageName**

скопировать настройки хранилищ у существующего **контейнера** containerName

**docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar cvf backup/backup.tar dbdata**

сделать архивированый backup директории dbdata контейнера dbstore

разбор команды:

**--volumes-from dbstore** - скопировать настройки хранилищ у существующего контейнера dbstore

**-v $(pwd):/backup** - в новом контейнере предназначенном для создания dump смонтировать дополнительную папку /backup (bind mounts) к текущей директории localhost

**tar cvf /backup/backup.tar dbdata** - выполнить консольную команду ubuntu внутри нового

контейнера - архивировать утилитой tar папку dbdata в папку backup в архив backup.tar

**docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar xvf /backup/backup.tar**

накатить backup

или (imho команда с ненужными заморочками, может для примера пригодится)

**docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu bash -c "cd /dbdata && tar xvf /backup/backup.tar --strip 1"**

разбор команды:

часть разобрана выше при создании backup, смотри туда

**bash -c "..."**

вызываем bash в контейнере, флаг -c говорит что команда будет составной, в кавычках передаем составную команду

**--strip 1** - говорит tar удалить первый уровень директории (dbdata) распаковываемого архива, т.к. в предыдущей командой мы уже переместились в эту директорию