????? ??????????? ? ?????????

Установка

Доп. ссылка

Добавить пользователя, который впоследствии будет запускать контейнеры

```
apt-get install sudo
usermod -aG sudo sergey
sudo apt-get update
```

Добавить сертификат и новое хранилище

(Вариант 1 - Debian)

```
sudo apt-get install ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

(Вариант 2 - Ubuntu)

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Установка docker и docker compose

(Вариант 1 Debian/Ubuntu)

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

(Вариант 2 Alt Linux, сертификаты ставить не надо)

```
sudo apt-get install -y docker-engine
sudo apt-get install -y docker-compose-v2
```

Дополнительные удобства

Запуск docker при старте системы

```
sudo systemctl enable docker
```

Добавить пользователя в группу docker

```
sudo usermod -aG docker $USER
```

Информация о текущем статусе docker

```
docker info
```

Для Alt Linux

```
sudo gpasswd -a USER docker
sudo systemctl enable --now docker
sudo init 6
```

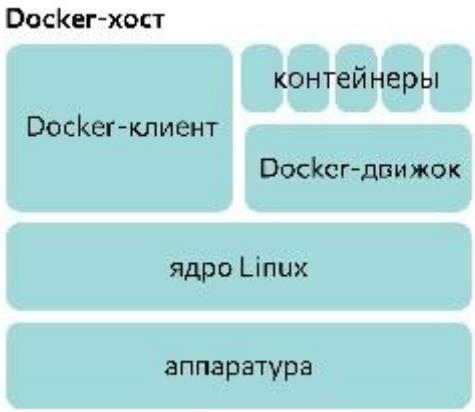
Размещение compose файлов.

Для dev: где угодно. Лучше определить, например ~/project_name

Для prod: /opt и /srv стандартизированы FHS (Filesystem Hierarchy Standard) для стороннего ПО. Я остановился на /srv/project_name/

Теория

Образы (Images): неизменяемые элементы, основанные как минимум на ядре и архитектуре. Контейнер: исполняемый образ. Назначение контейнера - запуск одного приложения. Аналогия: Образ=класс, контейнер=объект. Т.е. к образу в момент запуска добавляются параметры и он становится исполняемым контейнером, но изменения в контейнере не затрагивают образ. Контейнер в текущем состоянии можно превратить в образ, но после остановки (если не сохранить данные в файловой системе хоста или еще где-либо) все данные будут потеряны. Повторный запуск будет произведен как будто создался новый объект. В этом принципиальное отличие от виртуальной машины.



Может быть: архитектура ПК-ядро-образ-образ-образ-контейнер

Архитектура

Docker client -> Docker engine (daemon) -> containerd -> runc-> shim

runc легкая обертка libcontainer, задача - создание контейнеров. Для создании контейнера создается экземпляр runc, он создает контейнер, запускает и завершается. Контейнер передается shim. Shim сопоставляет процесс созданного контейнера с задачей в containerd, containerd логика выполнения контейнера, управления образами, томами, сетью, модульный, можно отключать

Пример процессов:

docker run --name ctr1 -it alpine:latest sh

- docker client преобразовывает в POST API запрос в Docker engine
- Запрос отправляется в сокет /var/run/docker.sock
- демон получает команду создания нового контейнера и передает вызов в containerd (CRUD-style API поверх gRPC)
- containerd преобразовывает Docker image и направляет запрос в runc для создания нового контейнера
- runc взаимодействует с ядром, создавая контейнер. Контейнер запускается как дочерний процесс runc. После запуска контейнера runc завершается.

Docker daemon может работать с сетью, non-TLS порт 2375, TLS port 2376

Контейнеры:

После запуска основным процессом становится приложение из параметров запуска

При отсутствии, используется приложение сконфигурированное в образе как точка запуска (Entrypoint)

Взаимоотношение между способами работы с Docker.

При работе с докером используются как минимум три способа (возможно больше) взаимодействий: прямое взаимодействие через консоль, dockerfile, docker compose. Терминология общая, но способы работы и результаты отличаются. Из-за того, что в целях упрощения консоли docker (групп команд docker ...) эти способы перемешаны, сначала возникает путаница. В статьях видимо по умолчанию считается, что читатель это отличие полностью осознал. Сравнение способов:

	Консоль	Dockerfile	docker compose

Мое понимание термина	Ввод команд настроек и запуска одного контейнера напрямую в консоли. Будет работать через bash скрипт, но удобнее так не делать.	Сохранение команд настроек одного контейнера в файл и запуск через консоль.	Настройка взаимодействия нескольких взаимосвязанных контейнеров при условии подготовленных образов. Запуск через консоль / планировщик
Применение	При тестовом запуске или для просмотра (корректировки) параметров работающего контейнера	Шаблон одного контейнера. Нужно при создании образа для последующего использования в compose.	Оркестрация образов.
Стадия использования	Тестовая, до MVP. Либо product для анализа проблем.	Тестирование, отладка одного образа. MVP - pre product.	Комплексный запуск сервиса. Pre product - product
Формат	Bash	ini	YAML
Точка использования	Преимущественно на ПК разработчика, при обучении технологии или для выявления проблем с работающим контейнером на всех стадиях.	ПК разработчика или тестовый сервер.	Тестовый / product сервер.
Необходимость серьезного изучения	Скорее администраторам.	Да, если для сервиса потребуется создание отдельных образов.	Да. Данный способ можно использовать для одного контейнера, для product все равно потребуется.*

* При сборке из чистых образов (типа debian для запуска nginx+...) требуется доп. установка и настройка ПО, а данный вопрос проще решать через dockerfile. К тому же существует например ansible, позволяющий решить ряд вопросов. В книгах для примеров часто используют консоль (быстро, акцент на одном изучаемом аспекте). А через ansible возможно настроить и запустить контейнер (и оркестр) без dockerfile/docker compose. Но в случае ansible для compose используется терминология compose, а вид команд другой.
Резюме: это одни яйца, только с разных концов. Нужно общее понимание происходящего. Хотя формат compose чуть предпочтительнее.

Revision #12
Created 23 June 2024 16:54:05 by Admin
Updated 12 May 2026 02:02:36 by Admin