Контейнеризация

Контейнеризация — метод *виртуализации*, при котором *ядро операционной системы* поддерживает несколько *изолированных экземпляров пространства пользователя* вместо одного.

Виртуализация – процесс создания вложенной изолированной вычислительной машины.

Ядро́ — центральная часть операционной системы, обеспечивающая приложениям координированный доступ к ресурсам компьютера.

Пользовательское пространство — адресное пространство виртуальной памяти операционной системы, отводимое для пользовательских программ.

Контейнер – в общем смысле это набор изолированных ресурсов, в которых может что-то выполняться.

Состоит из нескольких частей:

- Псевдо-ядра (чтобы контейнер мог управлять процессами, осуществлять запросы к ядру родительской ОС),
- пользовательская среда (интерпретатор и всё то, что нужно чтобы запускать код),
- настройки среды и службы родительской ОС, предоставляемые контейнеру

Отличия контейнера от полноценной виртуальной машины

Виртуальная машина (разметка для эмулируемой аппаратной части и установленная на неё ОС)	Контейнер
Полноценная операционная система со всеми функциями	Отдельная группа процессов, управляемых своими ресурсами. Нет традиционных для ОС компонентов: графической оболочки, драйверов и т.д.
Занимает гигабайты памяти	Занимает несколько мегабайт памяти
Загружается медленно	Загружается быстро
Полная изоляция процессов, файлов и служб	Обращается к ядру родительской ОС
Независимая от родительской ОС ядро	То же самое псевдо-ядро, что и у родительской ОС (чтобы обращаться к службам родительской ОС)

Применение контейнера: запуск кода в изолированной среде и обеспечения доступа к приложению, которое явилось следствием запущенного кода.

Поэтому контейнер содержит только то необходимое, которое нужно для запуска кода: псевдо-ядро (чтобы запустить код, управлять процессами, обращаться к ядру родительской ОС), интерпретатор (обработчик кода) и необходимые дополнения (библиотеки и т.д.)

Пример использования контейнера:

Необходимо запустить код сайта и обеспечить к сайту доступ клиентов. Предоставлен код.

Варианты решения:

1) Взять сервер, установить гипервизор, создать виртуальную машину, установить на неё ОС и в этой ОС запустить код сайта.

2) Взять ПК с ОС Windows, установить VirtualBox, создать виртуальную машину, установить на неё ОС и в этой ОС запустить код сайта.

3) Взять ПК с ОС Linux, установить ПО для создания контейнеров, создать контейнер с необходимыми внутренностями и запустить в нём сайт.

Взять ПК с ОС Linux, установить ПО для создания контейнеров, создать контейнер с необходимыми внутренностями и запустить в нём сайт.

1) Какое ПО позволяет разворачивать контейнеры?
2) Откуда взять готовый контейнер или как его создать самому?
3) Как запустить сайт в контейнере?

Docker — программное обеспечение для управления приложениями в средах с поддержкой контейнеризации, *контейнеризатор приложений.*

Команда для установки (в общем виде): **apt install docker**

Приложение docker - это запускаемая команда, которая обрабатывает все задачи управления для системы Docker.

Администраторы взаимодействуют с демоном dockerd через командную строку, запуская подкоманды команды docker. Например, вы можете создать контейнер с помощью команды docker run или просмотреть информацию о сервере с помощью команды docker info.

Подкоманда	Предназначение
`docker info`	Отображает сводную информацию о демоне
`docker ps`	Отображает запущенные контейнеры
`docker version`	Отображает обширную информацию о версии сервера и клиента
`docker rm`	Удаляет контейнер
`docker rmi`	Удаляет образ
`docker images`	Отображает локальные образы
`docker inspect`	Отображает конфигурацию контейнера (данные в формате JSON)
`docker logs`	Отображает стандартный вывод из контейнера
`docker exec`	Выполняет команду в существующем контейнере
`docker run`	Запускает новый контейнер
`docker pull/push`	Загружает образы из удаленного реестра или загружает образы в удаленный реестр
`docker start/stop`	Запускает или останавливает существующий контейнер
`docker top`	Отображает состояние контейнерного процесса

Откуда берутся контейнеры?

Контейнер это уже готовая небольшая виртуальная машина, в которой есть псевдо-ядро, какой-то интерпретатор и библиотеки. Псевдо-ядро выделяет ресурсы, запускает код с помощью интерпретатора.

Контейнеры создаются из образа контейнера (инструкции, по которым надо создать эту виртуальную машину и то, что в ней должно быть). Или используются уже «готовые» контейнеры, в которых есть и необходимые службы, и код для исполнения.

Чтобы создать контейнер, можно взять любую ОС, «извлечь» часть ядра, добавить к нему необходимый интерпретатор и т.д.

Но, проще взять готовый образ. У таких контейнеров уже есть какое-то псевдо-ядро ОС и внутреннее ПО. Для готовых контейнеров есть специальные репозитории. Процесс поиска готового контейнера похож на процесс поиска образа сборки какой-нибудь ОС.

Образ является шаблоном для контейнера. В него включены файлы, от которых зависят процессы, выполняемые в экземпляре контейнера, такие как библиотеки, двоичные файлы операционной системы и приложения.

По шаблону можно развернуть десятки и сотни одинаковых контейнеров, не нужно будет каждый раз прописывать вручную параметры для создания контейнера и то, что должно быть в нём.

Сетевые реквизиты

Доступ к контейнеру осуществляется по сетевым реквизитам.

Он обеспечивается стандартной настройкой сетевых реквизитов – адресом и портом доступа. Основная ОС, в которой установлен контейнер, становится упрощенным маршрутизатором, который получает трафик из внешнего мира и отдает его контейнеру. Или наоборот, отдает трафик контейнера внешнему миру.

Теперь, когда ПО для контейнеризации установлено, найден образ контейнера, его нужно развернуть. В самом базовом случае это можно сделать одной командой без параметров:

docker run deЬian

После ввода команды в ОС появится контейнер, в котором ядро Debian и некоторое окружение. Но кода там ещё нет

Как работать внутри контейнера?

При описанном выше запуске — запускается сам сервис, но мы сами не попадаем "внутрь".Для попадания "внутрь" — нужно переопределить команду старта образа.

Поэтому, вводится команда запуска, некоторые ключи (о них позже), реквизиты доступа (адрес и порт), обращение к переменным окружения (находятся в папке) и, самое главное, ash. Это указание запустить shell-оболочку.

```
docker run --rm -it -e CLICKHOUSE_ADDR=127.0.0.1:9000 kaktuss/clickhouse-udp-proxy ash
```

В этом контексте оболочка это интерпретатор команд для операционной системы контейнера.

После ввода команды оказываемся «внутри» контейнера и нам доступна оболочка ОС контейнера, к которой мы можем обращаться с помощью команд. Равносильно тому, что запустить Debian на обычной виртуальной машине и вводить какие-то команды.

Как код помещается в контейнер?

Нужно поместить код в родительскую ОС и дать контейнеру доступ к папке с кодом. После этого его можно переносить внутрь, редактировать и запускать.

Но, можно запустить код и не копируя его в контейнер. Для этого нужно тоже дать контейнеру права доступа к папке с кодом.

Отдельная история – контейнер с кодом. **Если код уже внутри, то он запускается после старта контейнера, дополнительные усилия почти не нужны.**

Стоит отметить, что контейнер только получает доступ к файлам родительской ОС для чтения и копирования. Своих файлов он не создаёт, они будут находиться только «внутри» контейнера.

После того, как сайт стал не нужен или возникла другая причина, по которой его нужно удалить, то это сделать просто.

Удаляется контейнер, и всё, что было в нем. Благодаря свойству изолированности все файлы, которые были нужны для работы сайта или являлись следствием его работы, хранятся в контейнере. Удаляем контейнер – удаляем всё.