ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Java-приложение в Docker

Петрелевич Сергей.

План занятия

- Контейнеризация в Linux
- Что такое Docker
- На основе чего сделать образ
- Как собрать образ
- Параметры Xmx, Xms
- Доступ к приложению через jmx
- Docker Compose

Контейнеризация в Linux

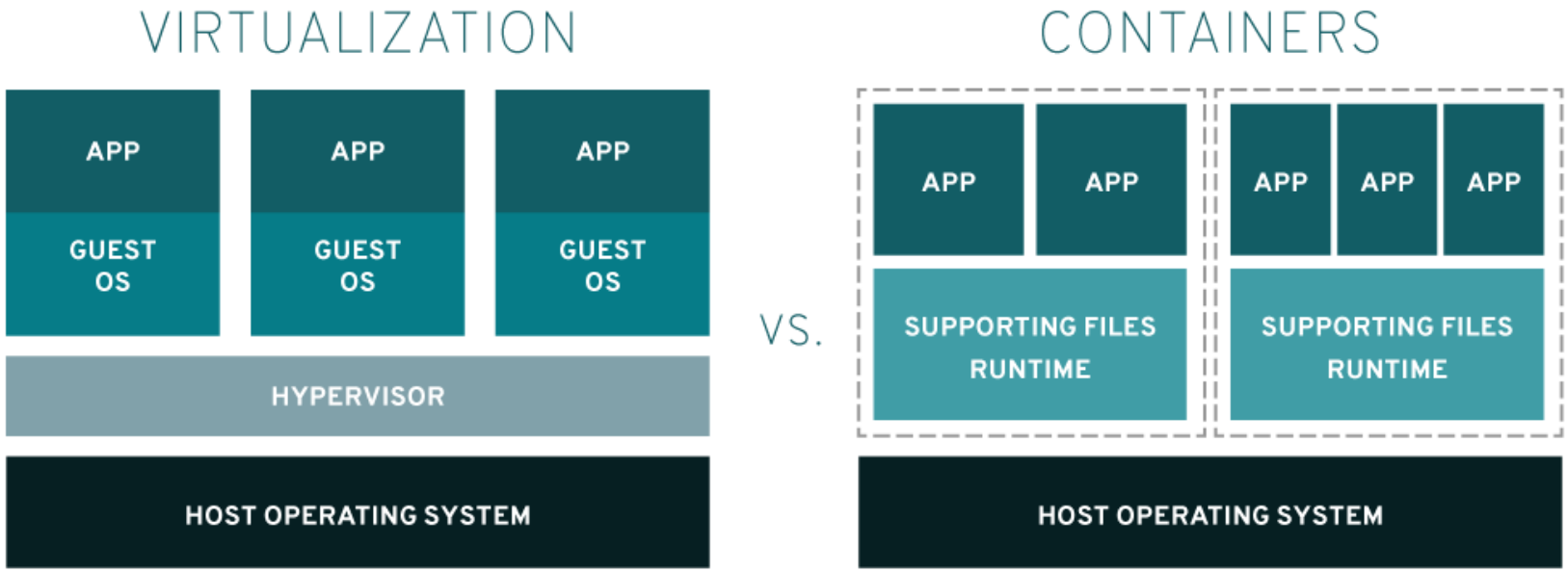
Linux containers (контейнеризация) это набор технологий, которые позволяют упаковывать приложения вместе с зависимостями и "средой выполнения".

Т.е. контейнер - это **типовой объект**, который содержит все нужное для работы приложения.

Контейнер – это просто своего рода архив?

Контейнеризация в Linux

Контейнер в Linux – это набор изолированных от остальной системы процессов. Но контейнер – это не виртуальная машина.



[Сравнение виртуализации и контейнеризации](#).

Популярная технология контейнеризации

LXC (Linux Containers)

LXC is the well-known and heavily tested low-level Linux container runtime.

Развивается с 2008 года.

LXC использует следующие возможности ядра операционной системы:

Kernel namespaces (ipc, uts, mount, pid, network and user)
Apparmor and SELinux profiles
Seccomp policies
Chroots (using pivot_root)
Kernel capabilities
CGroups (control groups)

Возможности ядра операционной системы

Namespaces

Технология, которая позволяет
разделить ресурсы ядра на изолированные группы

Примеры namespaces Linux:

pid namespace: Process isolation (PID: Process ID).
net namespace: Managing network interfaces (NET: Networking).
ipc namespace: Managing access to IPC resources.
mnt namespace: Managing filesystem mount points (MNT: Mount).
uts namespace: Isolating kernel and version identifiers.

Возможности ядра операционной системы

Control groups

Технология, которая позволяет
определить лимит приложения на использование ресурсов

https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

Как на них посмотреть:
ls /sys/fs/cgroup/

Docker

Docker Engine is the industry's de facto container runtime that runs on various Linux and Windows Server operating systems.

Docker creates simple tooling and a *universal packaging* approach that bundles up all application dependencies inside a container which is then *run* on Docker Engine.

Что Docker добавляет к технологии LXC:

- Portable deployment across machines (формат контейнера)
- Application-centric (оптимизация для разработки приложения)
- Automatic build (тулы для сборки образа)
- Versioning (git-похожее версионирование)
- Component re-use (переиспользование образов)
- Sharing (Docker Hub)
- Tool ecosystem (множество разных тулов)

Технологическая основа Docker

Namespaces

разделяет ресурсы ядра на изолированные группы

Control groups

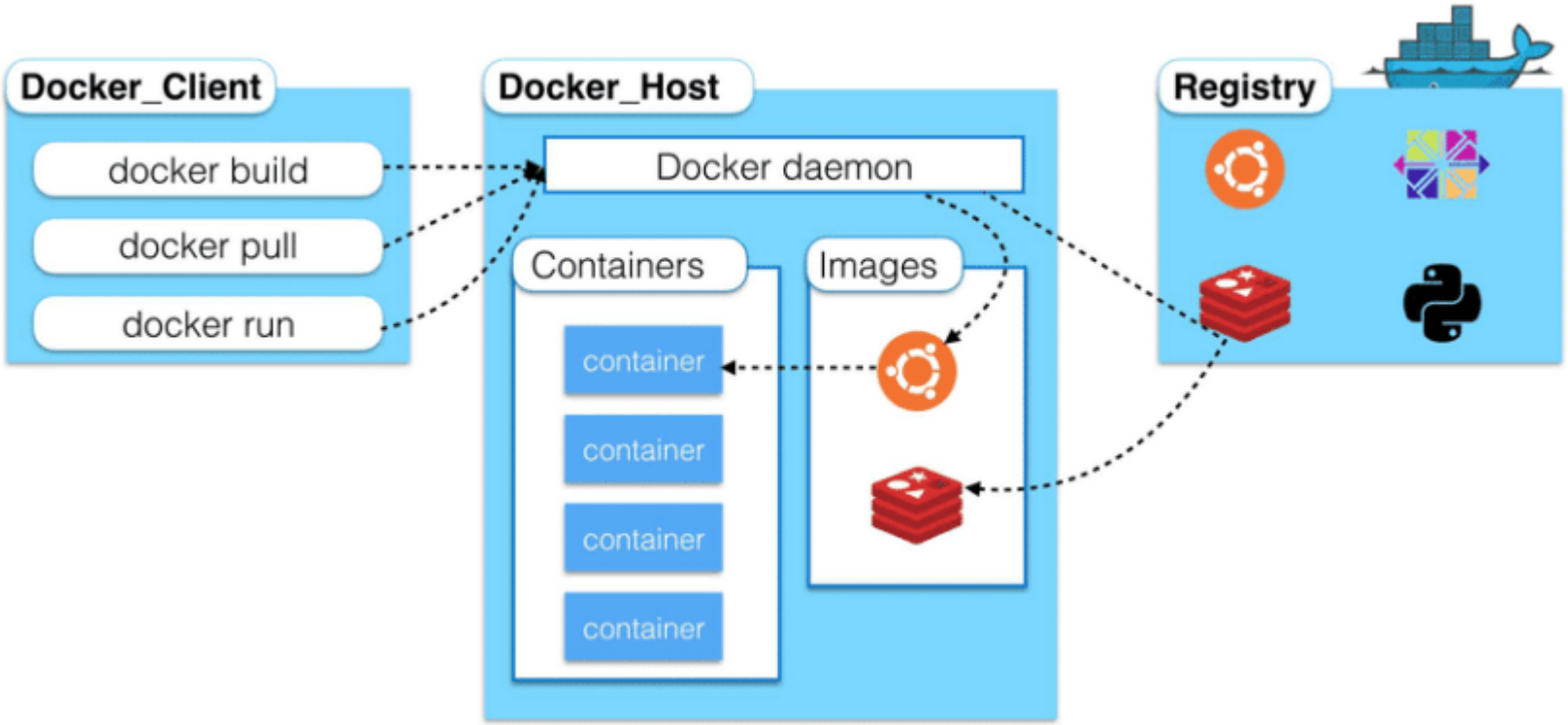
определяет лимит приложения на использование ресурсов

Union file systems

Union file systems (UnionFS) – файловые системы, которые управляются путем создания слоев. Это делает их быстрыми и легкими.

Docker, архитектура

https://docs.docker.com/engine/docker-overview/



Основные компоненты

Docker daemon

Docker client

Docker registries

в нем хранятся

IMAGES

из них создаются

CONTAINERS

На основе чего собрать образ

Нужен исходный образ с Java.

Возможные варианты:

- ~~openJDK~~
- Liberica
  bellsoft/liberica-openjdk-centos
  bellsoft/liberica-openjdk-alpine
  bellsoft/liberica-openjdk-alpine-musl
  tag: 11.0.3

Сравним размеры:

docker images

OTUS

Как собрать образ

Нам потребуется:

Dockerfile

pom-файл для maven-а

Собираем три образа для трех базовых образов
И проверяем, что приложение запускается.

docker run java-docker-centos
docker run java-docker-alpine
docker run java-docker-alpine-musl

Пробуем bash в контейнере
docker run -it java-docker-centos /bin/bash
docker run -it java-docker-alpine /bin/bash
docker run -it java-docker-alpine-musl /bin/bash

Параметры Xmx, Xms

Варианты запуска

1) Без ограничений

docker run java-docker-centos

2) Ограничения docker

docker run --memory=100m --memory-swap=100m --cpus 2 java-docker-centos

3) Ограничения -Xmx10g -Xms10g

причем -Xmx10g -Xms10g больше чем --memory=58m

Наблюдаем как приложение «закроется» контейнером

dmesg -T

*4) «*Возвращаемся» во времена java8 без «поддержки» контейнера

Java в Docker «видит» только ограничения по процессору.
На лимит по памяти реагирует слабо.

Что делать?

Утилита free в Docker

Из контейнера
docker run --memory=100m --memory-swap=100m --cpus 2 -it java-docker-centos /bin/bash

запускаем команду

free –h

Учим free жить в контейнере

Устанавливаем lxcfs

sudo dnf install lxcfs

LXCFS is a simple userspace filesystem designed to work around some current limitations of the Linux kernel.

sudo systemctl enable lxcfs.service

sudo systemctl status lxcfs.service

Повторяем опыт с free, но с «пробросом лимита»

docker run --memory=100m --memory-swap=100m --cpus 2 -it -v /var/lib/lxcfs/proc/meminfo:/proc/meminfo java-docker-centos /bin/bash

free -h

Запускаем приложение в контейнере

Запускаем приложение с «проброшенными лимитами» .

docker run --memory=100m --memory-swap=100m --cpus 2 -v /var/lib/lxcfs/proc/meminfo:/proc/meminfo java-docker-centos

-Xmx10g -Xms10g все еще больше *--memory=100m*

Команда free и java водят лимит как физическое ограничение.

Свалится ли программа при старте при попытке выделить хип больше физического лимита?

Запускаем приложение без контейнера

Если запускать Java-приложение с –Xmx –Xms больше, чем доступно в системе, получим примерно такую ошибку:

LibericaJDK 64-Bit Server VM warning: INFO: os::commit_memory(0x00007eed70000000, 107374182400, 0) failed; error='Not enough space' (errno=12)

Фактически, это же видим и при работе в контейнере, если вспомним, что лимит контейнера - это «лимит на использование».

Подключаем JMX

С этим все просто.

При сборке указываем стандартные параметры:

-Dcom.sun.management.jmxremote

-Dcom.sun.management.jmxremote.port=1026

-Dcom.sun.management.jmxremote.local.only=false

-Dcom.sun.management.jmxremote.authenticate=false

-Dcom.sun.management.jmxremote.ssl=false

При запуске контейнера пробрасываем порт

docker run -p1026:1026 java-docker

Обратите внимание на pid процесса java-приложения.

Docker compose, пример

https://docs.docker.com/compose/

Compose is a tool for defining and running multi-container Docker applications.

Посмотрим пример.

OTUS

Вопросы?

Опрос !!!

Пожалуйста, пройдите опрос.

OTUS

Спасибо за внимание!