Архитектура программных систем. Лекция 2 Контейнерная виртуализация

Михаил Пожидаев

21 февраля 2025 г.

Контейнерная виртуализация

Основные черты контейнерной виртуализации

1. Код исполняется нативно.
2. В гостевых системах ядро общее с хост-системой.
3. Каждый контейнер имеет свой IP-адрес.
4. Лимитируется память, процессорное время и пр.

Недостатки

Минусы контейнерной виртуализации

При сплошных плюсах технологии контейнерной виртуализации есть и некоторые минусы:

1. Слишком быстро копится overhead использования дискового пространства.
2. Сильная привязка к ядру хост-системы.
3. На Windows (равно как и в других ОС вообще) исполняется в виртуальной машине, что сводит на нет полезный эффект.
4. Не эмулирует недоступное оборудование.

Технологии

Реализация контейнерной виртуализации

Пространства имён (namespace)

Одна копия Linux может разделять на пространства имён следующие подсистемы: файловые системы, идентификаторы процессов, сетевые атрибуты, элементы межпроцессного взаимодействия, идентификаторы пользователей.

Контрольные группы(cgroups)

Позволяют лимитировать интенсивность использования таких ресурсов как: память, сетевой обмен, ввод/вывод, процессорное время.

Docker

От компании Google

Docker — система управления контейнерами на базе пространств имён и контрольных групп Linux.

1. Автоматически конфигурирует сеть для гостевых систем и хост-системы.
2. Использует каскадное объединение томов хранения файлов.
3. Поддерживает публичный репозиторий образов контейнеров.
4. Написана на языке Go.

Docker

Управление множеством контейнеров

- *Docker Compose* — сборка множества и локальный запуск контейнеров в отладочных целях;
- *Docker Sworm* —- распределённый запуск контейнеров на множестве хостов.
- *Docker Stack* — управление Docker Swarm в декларативном режиме.

Сервис

Как основная вычислительная единица

Сервис — множество контейнеров, запущенных из одного образа на одном или нескольких хостах.
Для конфигурирования сервиса необходимо определить:

- образ, на основе которого будут создаваться контейнеры;
- необходимые требования к вычислительным ресурсам и лимиты;
- дополнительные правила выбора хостов;
- подключаемые дисковые тома, сети и пр.

Оркестрация

Централизованное управление кластером

Дано:

- множество хостов-воркеров с известной конфигурацией;
- желаемая конфигурация сервисов.

В задачу оркестрации входит поиск наиболее удачного распределения контейнеров для запуска на множестве доступных хостов-воркеров. При этом утилита оркестрации должна отслеживать остановку контейнеров и выполнять их автоматический перезапуск в случае необходимости. Возможно динамическое изменений конфигурации кластера путём минимальных изменений.

Docker Compose

Пример конфигурации сервиса

```
mongodb:
  hostname: mongo1
  image: mongo:5.0
  command: mongod --replSet rs1
  ports:
    - 27017:27017
```

Docker Stack

Пример конфигурации сервиса

```
kafka:
  image: confluentinc/cp-kafka:latest
  depends_on:
    - zookeeper
  deploy:
    replicas: 1
    resources:
      limits:
        memory: 1088M
      reservations:
        memory: 1088M
    placement:
      constraints: [node.role == manager]
```

Источники образов

Что мы можем запустить в сервисах?

1. Образы могут быть получены из Docker Hub.
2. Образы могут быть получены из локальных реестров образов.
3. В случае Docker Compose образы могут быть собраны на локальной системе.

Пример имени образа из реестра образов Яндекса: `cr.yandex/RepoId/db`.

Dockerfile

Основные команды сборки образа

- `FROM` — имя базового образа;
- `ADD` и `COPY` — добавить локальный файл в образ;
- `RUN` — выполнить команду внутри образа;
- `CMD` — установить команду для запуска контейнера;
- `WORKDIR` — установить рабочий каталог.

Dockerfile

Пример файла

```
FROM node:latest
WORKDIR /app
EXPOSE 80
RUN npm install formidable
RUN npm install send
COPY . .
CMD [ "nodejs", "main.js" ]
```

Реестр образов — это централизованное хранилище, в котором хранятся образы контейнеров, содержащие все необходимые компоненты для запуска. Использование реестра упрощает процесс разработки, тестирования и развёртывания приложений, а также обеспечивает централизованное управление версиями и безопасностью.

1. **Централизованное управление:** упрощает контроль версий и развёртывания приложений.
2. **Масштабируемость:** обеспечивает быстрое развёртывание и масштабирование приложений.
3. **Эффективность:** ускоряет процесс разработки и тестирования приложений.

Популярные реестры образов контейнеров:

- **Docker Hub:** официальный реестр Docker.
- **Google Container Registry (GCR):** реестр контейнеров от Google Cloud.
- **Amazon Elastic Container Registry (ECR):** реестр контейнеров от Amazon Web Services, который интегрируется с другими сервисами AWS.
- **Quay:** реестр контейнеров оот Red Hat.

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/arch.html
E-mail: msp@luwrain.org
Канал в Телеграм: @MarigostraRu