Docker

18 марта 2026 г.

Что мы имеем после прошлой лекции

В GitHub Actions настроен CI/CD пайплайн :

- Pull request → запуск тестов; Git push → запуск тестов, автоматический деплой

Но:

- Локальная разработка — боль
- Нет разграничения dev/prod
- Проблема «серверов-снежинок»

Локальная разработка

1. Runtime зависимости: базы данных, внешние сервисы, библиотеки
2. Разные Linux дистрибутивы: Ubuntu, CentOS, Arch и др.
3. Разные операционные системы: Windows, macOS, Linux

Работа с окружениями dev/prod

1. Один сервер с nginx и для каждого окружения свой конфиг
2. Каждому окружению свой сервер

Демонстрация: деплой в dev и prod

Задача: настроить деплой пайплайн для разных окружений

Что посмотрим:

- deploy-dev.yml и deploy-prod.yml
- GitHub Environments
- *ps -C nginx -o rss= | awk '{sum+=$1} END {print sum/1024}'*

Демо-репозиторий:
https://github.com/prafdin/devops-demo-website/tree/envs-demo

Сервера-снежинки

- Каждый сервер — особенный
  - *nginx -v*
  - *curl -I*
- Configuration drift

Решение — контейнеризация

- Удобная работа с окружением и зависимостями
- Изоляция процессов
  - Удобная работа с несколькими экземплярами одного приложения
  - Безопасность
  - Контроль использования ресурсов
- Стандартизация процесса развертывания

Что такое контейнеризация?

Контейнеризация — технология изоляции приложений с их зависимостями в отдельные исполняемые единицы.

Контейнер — процесс или группа процессов, запущенные в изолированном окружении.

Изоляция на уровне:

- **Процессов** — у каждого контейнера своё дерево процессов
- **Сети** — у каждого контейнера свои сетевые интерфейсы, настройки маршрутизации
- **Ресурсов** — cpu, ram, IO
- **Файловой системы** — у каждого контейнера своя файловая система
- **...** — а также много других механизмов ядра

Виртуальные машины vs Контейнеры

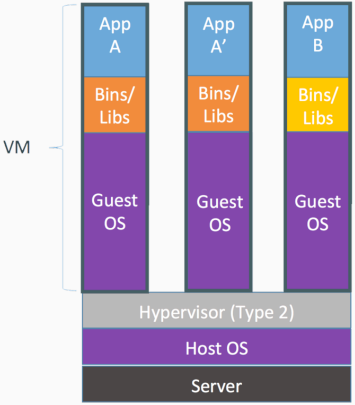
Виртуальные машины

- Полная ОС для каждого приложения
- Большой overhead
- Медленный старт
- Изоляция на уровне железа/гипервизора

Контейнеры

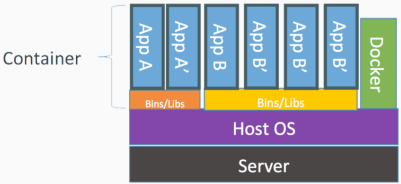
- Общее ядро ОС
- Минимальный overhead
- Быстрый старт (секунды)
- Изоляция на уровне процессов

Виртуальные машины vs Контейнеры



Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



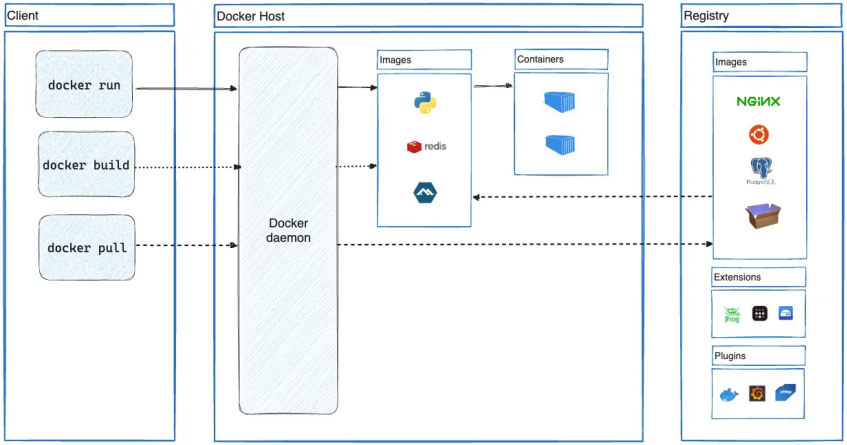
https://dzone.com/articles/evolution-of-linux-containers-future

Способы работы с контейнерами

Существует несколько технологий контейнеризации:

- **Docker** — самая популярная платформа контейнеризации
- **Podman** — альтернатива Docker без демона
- **LXC/LXD** — системные контейнеры

Устройство docker



https://docs.docker.com/get-started/docker-overview

Основные понятия Docker

Ключевые абстракции Docker:

- **Container** — процесс или группа процессов, запущенные в изолированном окружении
- **Image** — шаблон контейнера, содержащий снимок файловой системы и метаданные контейнера
- **Dockerfile** — инструкции для сборки образа (image)
- **Registry** — серверное приложение для хранения и передачи образов

Принцип работы:

1. Описываем как собрать образ в Dockerfile
2. Собираем образ из Dockerfile
3. Запускаем контейнер из образа

Демонстрация: переход на docker в CI/CD пайплайне

Задача: собирать и распространять приложение как образ docker контейнера

Что посмотрим:

- Сборка и деплой через Docker контейнеры
- Запуск проекта в Windows

Демо-репозиторий:
https://github.com/prafdin/devops-demo-website/tree/container-demo

Материалы по теме

- https://martinfowler.com/bliki/SnowflakeServer.html
- https://habr.com/ru/articles/556868/
- https://iximiuz.com/en/categories/?category=Containers