Docker.

Введение.

Docker — это платформа для создания, развертывания и управления изолированными приложениями.

Основная идея Docker заключается в том, чтобы позволить разработчикам создавать приложения в изолированных контейнерах, которые могут быть легко перенесены между различными средами (например, с разработки в продакшн) и запускаться одинаково везде.

Основные аспекты Docker помогают определить его значимость в программировании;

Определение, принцип работы.

Docker — это открытая платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковывать приложение и все его зависимости в единый стандартный блок, который может работать везде, где работает Docker. Docker основан на концепции контейнеризации, которая позволяет запускать приложения и их зависимости в изолированных средах. Принципы работы можно разбить на ключевые компоненты и процессы.

Принципы работы

1. **Образ (Image)**: статический файл, содержащий всё необходимое для запуска контейнера: код, библиотеки, зависимости, инструменты и конфигурации. Образы создаются на основе Dockerfile, который содержит инструкции для сборки образа.
2. **Контейнер (Container)**: запущенный экземпляр образа. Контейнеры изолированы друг от друга и от хоста, но могут взаимодействовать через определенные каналы (например, сети и тома).
3. **Docker Engine**: движок, который управляет контейнерами. Он включает в себя демона (docker daemon), который выполняет команды Docker CLI и API.

Виды Docker.

Docker предоставляет **несколько видов продуктов**, которые ориентированы на разные типы пользователей и случаи использования. Основные из них — это **Docker CE (Community Edition)** и **Docker EE (Enterprise Edition)**. Каждый из них имеет свои особенности и предназначен для различных потребностей.

Основными характеристиками которых являются:

- безопасность, целостность и доступность;
- поддержка сообществом программистов/специалистов;
- различный список версий;
- управление версионностью;
- простота установки;
- и другие;

Виды Docker. Docker CE.

Docker CE — это бесплатная версия Docker, предназначенная для индивидуальных разработчиков, небольших проектов и тех, кто хочет экспериментировать с контейнеризацией. Она доступна с открытым исходным кодом и предоставляет все основные возможности Docker для разработки, тестирования и развертывания контейнеров.

Основные характеристики Docker CE:

1. **Бесплатность и доступность:** Docker CE доступен бесплатно и может быть установлен на различных операционных системах, включая Windows, macOS и Linux.
2. **Обновления и новые функции:** Docker CE получает регулярные обновления и новые функции, которые разрабатываются сообществом и Docker Inc.
3. **Поддержка сообществом:** Поддержка осуществляется через форумы, GitHub и другие ресурсы сообщества.
4. **Легкость установки:** Простая установка и настройка на различных платформах.
5. **Версии:**
    - **Stable:** Стабильная версия, обновляется раз в квартал.
    - **Edge:** Новая версия, обновляется каждый месяц, включает последние функции и улучшения.

Виды Docker. Пример использования - Docker CE.

Docker CE - бесплатная платформа, что имеет широкий спектр возможностей и применение. Безопасность и доступность позволяет ей стабильно и надежно работать, но реализация функциональной безопасность критически ограниченно по-сравнению с платной платформой Docker EE. Версии помогают выбрать нужный формат для успешной дальнейшей работы с Docker CE.

Пример использования Docker CE:

Разработчик может использовать Docker CE для создания и тестирования контейнеризированных приложений на своей рабочей станции, а затем легко перемещать эти контейнеры между различными средами (например, с разработки в тестирование и продакшн).

Виды Docker. Docker EE.

Docker EE — это платная версия Docker, предназначенная для корпоративных клиентов. Она включает в себя все возможности Docker CE, а также дополнительные функции безопасности, управления и поддержки. Docker EE предназначен для организаций, которым требуется высокий уровень надежности, безопасности и масштабируемости.

Основные характеристики Docker EE:

1. **Коммерческая поддержка:** Docker EE включает коммерческую поддержку, предоставляемую Docker Inc., с доступом к профессиональным инженерам поддержки и консультационным услугам.
2. **Дополнительные функции безопасности:**
    - Docker EE включает в себя функции для обеспечения безопасности контейнеров, такие как:
        - **Image Signing:** Подписание образов для проверки их подлинности.
        - **Role-Based Access Control (RBAC):** Управление доступом пользователей на основе ролей.
        - **Secrets Management:** Управление конфиденциальными данными, такими как пароли и ключи API.
3. **Оркестрация (Управление контейнерными средами.) и управление:** Встроенная поддержка оркестрации контейнеров с помощью Docker Swarm и Kubernetes, включая возможности мониторинга и масштабирования.
4. **Интеграция с существующими системами:** Интеграция с популярными инструментами CI/CD (Continuous Integration/Continuous Deployment), системами управления конфигурациями и другими корпоративными системами.
5. **Управление на уровне предприятия:** Централизованное управление и контроль над контейнерными приложениями, включая мониторинг, логирование и анализ.

Виды Docker. Пример использования - Docker EE.

Docker EE - платная платформа, которая имеет ряд преимуществ над EE. Коммерческая поддержка усиливает надежность платформы. Безопасность и доступность позволяет ей стабильно и надежно работать - функциональная безопасность значительно лучше, чем у Docker CE.

Пример использования Docker EE:

Большая корпорация может использовать Docker EE для развертывания и управления своими контейнеризированными приложениями в продакшн среде, где необходим высокий уровень безопасности, надежности и поддержка корпоративных стандартов.

Docker EE. Функции обеспечения безопасности;

Image Signing (Подписание образов)

Описание: Процесс подписания Docker-образов с помощью криптографических ключей для обеспечения их подлинности и целостности.

Назначение: Гарантирует, что образы не были изменены и поступают из доверенного источника.

Преимущества:

- Повышает безопасность при развертывании контейнеров.
- Защищает от использования поддельных или скомпрометированных образов.

Role-Based Access Control (RBAC)

Описание: Система управления доступом, где права пользователей определяются их ролями в организации.

Назначение: Контролирует, какие действия могут выполнять пользователи и группы пользователей, основываясь на назначенных им ролях.

Преимущества:

- Повышает безопасность путем ограничения доступа.
- Упрощает управление правами пользователей в крупной организации.

Secrets Management (Управление секретами)

Описание: Процесс безопасного хранения, распределения и управления конфиденциальными данными, такими как пароли, токены и ключи API.

Назначение: Защищает конфиденциальные данные от несанкционированного доступа.

Преимущества:

- Обеспечивает безопасность чувствительной информации.
- Упрощает управление и обновление секретов в контейнерных средах.

Сравнение CE vs EE;

Характеристика	Docker CE	Docker EE
Стоимость	Бесплатно	Платно
Целевая аудитория	Разработчики, малые проекты	Корпоративные клиенты
Поддержка	Сообщество	Коммерческая поддержка
Функции безопасности	Ограниченные	Расширенные (RBAC, Image Signing)
Управление контейнерными средами	Docker Swarm, Kubernetes (частично)	Docker Swarm, Kubernetes (полная)
Обновления	Часто	Редко, стабильные
Управление	Локальное	Централизованное

DockerFile. Определение, расширение и формат.

Dockerfile — это текстовый файл, содержащий набор инструкций, которые Docker использует для автоматизации сборки образов контейнеров. Каждая инструкция в Dockerfile описывает шаг, необходимый для создания образа, начиная с базового образа и заканчивая конфигурацией приложения.

Расширение и формат:

- **Расширение:** По умолчанию, Dockerfile не имеет расширения и обычно называется просто Dockerfile. Однако, для удобства и ясности в некоторых случаях могут использоваться кастомные имена с расширением .dockerfile (например, Dockerfile.web).
- **Формат:** текстовый файл, который следует специфическому синтаксису и структуре Dockerfile-инструкций.

DockerFile. Основные инструкции/функции.

- **FROM:** Указывает базовый образ, с которого начинается сборка.
- **WORKDIR:** Задает рабочую директорию внутри контейнера.
- **COPY:** Копирует файлы и директории с хоста в контейнер.
- **RUN:** Выполняет команды в контейнере на этапе сборки.
- **CMD:** Определяет команду, которая будет выполнена при запуске контейнера.
- **EXPOSE:** Открывает порты, которые будут использованы приложением.
- **ENV:** Устанавливает переменные окружения.
- **ENTRYPOINT:** Задает команду, которая будет выполнена при запуске контейнера, и позволяет добавлять параметры через docker run.

Сборка образа: `docker build -t my-python-app .`

Запуск контейнера: `docker run -d -p 5000:5000 my-python-app`

Команда docker build читает Dockerfile и создает образ, используя инструкции в файле. Параметр -t задает тег для нового образа, а . указывает текущую директорию, где находится Dockerfile.

Команда docker run создает и запускает контейнер из образа my-python-app. Параметр -d запускает контейнер в фоновом режиме, а -p пробрасывает порт 5000 на хост-систему.

DockerFile. На практике;

```
# Используем официальный образ Python
FROM python:3.8-slim

# Устанавливаем рабочую директорию
WORKDIR /app

# Копируем файлы проекта в контейнер
COPY . /app

# Устанавливаем зависимости
RUN pip install -r requirements.txt

# Открываем порт 5000
EXPOSE 5000

# Указываем команду для запуска приложения
CMD ["python", "app.py"]
```

Docker-Checker;

Docker-Checker - понятие связано с проверками созданного docker;

- **Проверка работы контейнера:**
  - Открыв браузер можно перейти по адресу `http://localhost:5000`.
  - Вы должны увидеть сообщение "Hello, World!". (для разные проектов разное!)
- **Просмотр работающих контейнеров:** `docker ps`
- **Остановка контейнера:** `docker stop <container_id>`
- **Удаление контейнера:** `docker rm <container_id>`

Docker-Compose;

Docker Compose — это инструмент для определения и управления многоконтейнерными Docker-приложениями. Используя файл docker-compose.yml, вы можете описывать сервисы, сети и тома, необходимые для вашего приложения.

```
version: '3'
services:
 web:
   image: app
   ports:
     - "5000:5000"
 redis:
   image: "redis:alpine"
```

Преимущества:

- Упрощает управление многоконтейнерными приложениями.
- Легко запускает и останавливает все связанные контейнеры одной командой (docker-compose up и docker-compose down).

Docker-swarm;

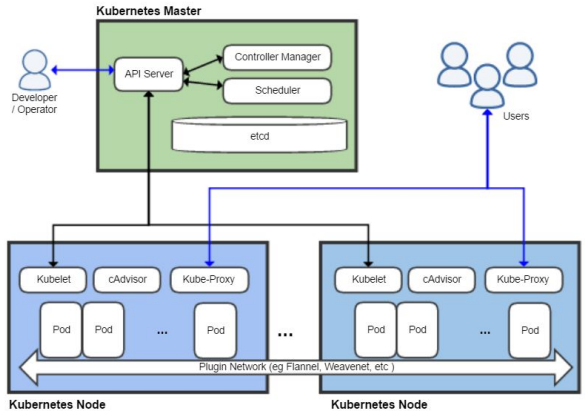
Docker Swarm — это встроенная система оркестрации контейнеров, предоставляемая Docker. Она позволяет пользователям создавать и управлять кластером Docker-движков (так называемым Swarm), которые работают вместе как единое целое. С помощью Docker Swarm можно развертывать контейнерные приложения, обеспечивать их масштабирование, высокую доступность и управление состоянием.

Преимущества:

- Простота настройки и интеграция с Docker CLI.
- Поддержка масштабирования сервисов и управления состоянием.

Kubernetes;

Kubernetes — это более мощная и широко используемая система оркестрации контейнеров, поддерживающая автоматическое развертывание, масштабирование и управление контейнерными приложениями.



Преимущества:

- Широкая экосистема и поддержка множества функциональностей.
- Поддержка управления состоянием и автоматического масштабирования.

Сети/Тома в Docker;

Сети:

- Docker позволяет создавать пользовательские сети, чтобы контейнеры могли общаться друг с другом.
- *Типы сетей включают bridge, host и overlay.*
    a. **Bridge** — это стандартная сеть по умолчанию для контейнеров на одном хосте. Простота настройки и изоляция контейнеров на уровне хоста.
    b. **Host** — это сеть, в которой контейнер использует сетевой стек хостовой машины. Высокая производительность за счет отсутствия изоляции сетевого стека.
    c. **Overlay** — это сеть, которая объединяет контейнеры, работающие на разных хостах в Swarm-кластере. Поддержка распределенных приложений. Безопасное соединение между узлами.

Тома:

- Docker использует тома для хранения данных, которые необходимо сохранять между запусками контейнеров.
- Тома могут быть локальными или удаленными, управляемыми сторонними плагинами.

Docker - хорошая практика использования;

Создание легковесных образов:

- Используйте минимальные базовые образы.
- Сократите количество слоев, объединяя команды RUN в единый ресурс.

Безопасность:

- Регулярно обновляйте образы и зависимости системы.
- Используйте подписанные и подлинные образы.
- Применяйте принципы минимизации привилегий для контейнеров.

Мониторинг и Логирование:

- Включите мониторинг контейнеров с помощью инструментов, таких как Prometheus и Grafana.
- Организуйте централизованное логирование с использованием стека ELK (Elasticsearch, Logstash, Kibana).

Микросервисная архитектура:

- Docker идеально подходит для микросервисной архитектуры, позволяя легко развертывать и управлять множеством микросервисов.