Слайд 1

Слайд 1

Слайд 1

Контейнеризация и Docker

Слайд 2

Что такое контейнеризация? Контейнеризация - это процесс упаковки приложения и его зависимостей в изолированный контейнер Контейнер - исполняемая единица программного обеспечения, включающая код, runtime, системные инструменты, библиотеки и настройки

Слайд 3

Особенности Docker Изолированность: контейнеры работают независимо от других процессов на хосте Портативность: контейнер можно запустить на любой платформе, которая поддерживает Docker Легковесность: контейнеры используют общий ядро операционной системы, что уменьшает размер и увеличивает производительность

Слайд 4

Применения Docker Упрощение развертывания приложений: Docker обеспечивает консистентную среду выполнения приложения на разных системах Масштабирование: Docker позволяет горизонтально масштабировать приложения, добавляя или удаляя контейнеры Управление зависимостями: Docker обеспечивает изолированное окружение для каждого приложения, что облегчает управление зависимостями и избегание конфликтов

Слайд 5

Docker Compose и оркестрация контейнеров Docker Compose - инструмент для определения и управления множеством Docker контейнеров Позволяет определить множество сервисов, настройки сети и другие параметры в едином файле YAML Облегчает запуск, остановку и масштабирование группы контейнеров в рамках одного проекта

Слайд 6

Пример использования Docker Compose

```
version: '3'
services:
  web:
    build: .
    ports:
      - 80:80
    volumes:
      - ./app:/app
    depends_on:
      - db
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: secret
```

Слайд 7

Примеры использования Docker * Развертывание веб-приложений в контейнерах * Тестирование приложений в изолированной среде * Микросервисная архитектура: запуск независимых сервисов в отдельных контейнерах * Continuous Integration/Continuous Deployment (CI/CD): автоматическое развертывание кода в контейнерах

Слайд 8

Выводы Docker позволяет упростить развертывание и управление приложениями Контейнеризация обеспечивает изолированное окружение для приложений Docker Compose позволяет оркестрировать группы контейнеров в рамках одного проекта

Слайд 1

Работа со сетью и томами в Docker

Слайд 2

Работа с сетью в Docker Docker предоставляет несколько вариантов работы с сетью: * По умолчанию контейнеры могут взаимодействовать только через локальный сетевой стек хоста * Можно создавать свои пользовательские сети, которые обеспечивают изоляцию и масштабируемость * Docker также поддерживает встроенные сети, такие как bridge (мост) и host (хост) * Можно связывать контейнеры, чтобы они могли взаимодействовать друг с другом

Слайд 3

Создание пользовательской сети в Docker Команда для создания пользовательской сети:

```
docker network create <network-name>
```

Можно указать дополнительные параметры, такие как тип сети или подсеть

Слайд 4

Связывание контейнеров в Docker Контейнеры могут быть связаны с помощью параметра –link или через пользовательские сети Пример:

docker run –name my-app –link redis-container:redis my-app-image

Слайд 5

Работа с томами в Docker Тома в Docker позволяют сохранять данные между запусками контейнера Docker позволяет использовать различные типы томов: Volumes (только для Docker): абстракция над файловой системой контейнеров Bind mounts: монтирование хостовых директорий в контейнер TMPFS mounts: монтирование временной файловой системы в контейнер

Слайд 6

Создание и использование томов в Docker Команда для создания и подключения тома:

docker volume create

docker run -v : my-app-image Можно также использовать полный путь к директории на хосте вместо

Слайд 7

Пример использования томов в Docker

```
version: '3'
services:
  db:
    image: mysql:5.7
    volumes:
      - db-data:/var/lib/mysql
  app:
    image: my-app-image
    volumes:
      - app-code:/app
volumes:
  db-data:
  app-code:
```

Слайд 8

Выводы Работа с сетью в Docker позволяет контейнерам взаимодействовать друг с другом Создание пользовательской сети и связывание контейнеров обеспечивает гибкость и изоляцию Тома позволяют сохранять данные между запусками контейнеров и облегчают работу с файлами в Docker