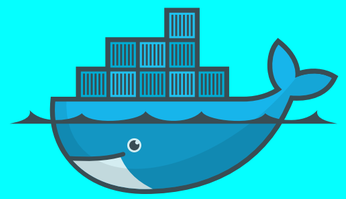
Вопросы и ответы на собеседование по Docker



(16-30)

16) Как узнать список работающих контейнеров Docker?

Для того чтобы узнать список работающих контейнеров Docker, вы можете использовать команду docker ps. Вот как это работает:

1. **Вывести список активных контейнеров:**
   - **docker ps**

Эта команда отобразит список контейнеров, которые в данный момент запущены.

2. **Вывести список всех контейнеров, включая остановленные**:

Если вам нужен список всех контейнеров, включая остановленные, вы можете использовать флаг **-a**:

   - **docker ps -a**

Это покажет вам полный список контейнеров на вашей системе, вне зависимости от их состояния.

17) Какой Docker-сетевой режим используется по умолчанию?

По умолчанию Docker использует **сетевой режим "bridge"** (или также известный как "**docker0**"). Этот режим позволяет контейнерам быть частью собственной изолированной сети, и они могут взаимодействовать друг с другом и с хостовой системой.

Сетевой режим "bridge" создает виртуальный **Ethernet**-адаптер на хостовой системе и назначает каждому контейнеру уникальный **IP-адрес** в этой виртуальной сети. Таким образом, контейнеры могут свободно общаться друг с другом по IP-адресам, а также могут использовать **NAT (Network Address Translation)** для доступа к внешним сетям, включая Интернет.

Этот сетевой режим обеспечивает изоляцию контейнеров друг от друга, что может быть полезно для безопасности и стабильности, и он является подходящим выбором для большинства сценариев.

Однако Docker также поддерживает различные сетевые режимы, включая "**host**", "**none**" и "**overlay**", которые могут быть настроены в зависимости от требований вашего приложения и окружения.

18) Какой тип приложений больше подходит для контейнеров Docker: с хранением состояния (stateful) или без хранения (stateless)?

Приложения без хранения состояния **(stateless) больше** подходят для работы в Docker, чем приложения с хранением (**stateful**). Мы можем создать контейнер для нашего приложения и принять некоторые его настройки. Таким образом мы можем запускать один и тот же контейнер с разными настройками для различных окружений.
Если мы не будем хранить состояние, сможем использовать один и тот же образ в разных сценариях. Также такие приложения проще масштабировать при их работе в контейнерах Docker.

19) Приведите необходимые шаги для развертывания докеризированного приложения, сохраненного в репозитории Git

Шаги, необходимые для развертывания приложения зависят от окружения, основной процесс развертывания будет таким:

- Сборка приложения с использованием Docker build в каталоге с кодом приложения
- Тестирование образа
- Выгрузка образа в Registry
- Уведомление удаленного сервера приложений, что он может скачать образ из Registry и запустить его
- Перестановка порта в прокси HTTP(S)
- Остановка старого контейнера

20) Чем Docker отличается от остальных технологий контейнеризации?

Docker — одна из последних разработок в контейнеризации, он стал одной из наиболее популярных. Docker, созданный в облачную эру, сделал возможным использование новых функций, ранее отсутствующих в старых технологиях контейнеризации. Самая крутая функция Docker — это работа с использованием любой инфраструктуры, неважно, у вас дома, либо в облаке.

Посредством Docker все больше приложений могут работать на старых серверах, также с его помощью можно упаковывать и поставлять программы. Существует также **DockerHub**, **Registry** для контейнеров, откуда можно легко и просто скачать образы контейнеров для использования. Еще более интересная функция — общие контейнеры для таких приложений. Также Docker хорошо документирован, что делает его лучше остальных технологий контейнеризации.

21) Если вы остановите контейнер — потеряете данные?

Если вы остановите контейнер в Docker, данные в контейнере **не будут автоматически удалены**, но они будут недоступны до тех пор, пока контейнер не будет снова запущен. Данные в контейнерах Docker обычно хранятся в слое **файловой системы контейнера**, который может быть сохранен даже после остановки контейнера.

Чтобы полностью потерять данные, необходимо удалить контейнер с опцией "**--rm**" при его создании, или удалить контейнер вручную после его остановки.

Опция "**--rm**" при создании контейнера автоматически удаляет контейнер после его завершения.

Чтобы сохранить данные в контейнере и обеспечить их доступность после остановки и удаления контейнера, рекомендуется использовать Docker **Volumes**. Docker Volumes представляют собой **постоянное хранилище данных**, которое может быть подключено к контейнеру. Это позволяет сохранять данные и обеспечивать их доступность даже после удаления контейнера.

Read & Write Code

22) Как выполняется мониторинг Docker в производственных окружениях?

Для мониторинга Docker-контейнеров в производственных окружениях существует несколько инструментов и методов. Вот некоторые из них:

- **Docker Logging**: Docker предоставляет встроенную поддержку для логирования контейнеров. Вы можете настроить контейнеры так, чтобы они отправляли свои логи в стандартный вывод (**stdout**) и стандартный поток ошибок (**stderr**), и затем перенаправлять эти логи в системы мониторинга или лог-агрегаторы.
- **Docker Stats**: Вы можете использовать команду docker stats для отслеживания использования ресурсов (CPU, память, сеть) контейнеров в реальном времени.
- **Системы мониторинга и логирования**: Для более продвинутого мониторинга и управления контейнерами Docker в производстве часто используются сторонние системы мониторинга и логирования, такие как **Prometheus, Grafana, ELK (Elasticsearch, Logstash, Kibana), Splunk** и многие другие. Эти инструменты позволяют собирать, анализировать и визуализировать данные о работе контейнеров, а также предоставляют мощные инструменты для определения аномалий и управления производственными окружениями.
- **Оркестраторы**: Оркестраторы, такие как **Kubernetes** и **Docker Swarm**, также предоставляют средства для мониторинга и управления контейнерами в кластере. Они могут автоматически масштабировать контейнеры, управлять нагрузкой и обеспечивать высокую доступность.
- **Мониторинг состояния контейнеров**: Множество инструментов и служб позволяют мониторить состояние контейнеров и реагировать на неполадки. Примеры включают Healthchecks, который позволяет определять, насколько здоров контейнер, и отказывать в балансировке нагрузки на него, если что-то идет не так.
- **Автоматическое обнаружение и мониторинг**: Сервисы, такие как **Consul**, могут автоматически обнаруживать контейнеры в сети и предоставлять данные о состоянии и мониторинге.

Для обеспечения высокой доступности и надежности в производственных окружениях важно комбинировать различные инструменты и методы мониторинга Docker, чтобы быстро обнаруживать и реагировать на проблемы.

Read & Write Code

23) Поясните разницу между docker run и docker create

docker run и **docker create** - это две команды в **Docker CLI**, каждая из которых выполняет разные действия при работе с контейнерами. Вот их разница:

- **docker run**:
  - Эта команда создает новый контейнер и запускает его сразу после создания.
  - Команда **docker run** объединяет два шага: создание контейнера и запуск его исполнения.
  - При использовании docker run, вы можете указать опции и параметры, такие как порты, переменные окружения, идентификаторы сетей и другие, которые будут применены к создаваемому и запускаемому контейнеру.
  - Пример: **docker run -d -p 8080:80 myapp:latest** создаст и запустит контейнер на основе образа **myapp:latest**, прокинет порт **8080** и привяжет его к порту **80** внутри контейнера.
- **docker create**:
  - Эта команда создает новый контейнер, **но не запускает его**. Контейнер остается в состоянии "**остановлен**".
  - После создания контейнера с помощью docker create, вы можете использовать команду **docker start**, чтобы запустить контейнер в будущем.
  - Команда docker create может быть полезной, если вы хотите подготовить контейнер, но не хотите его запускать немедленно. Это может быть полезно, например, при создании контейнера вручную или при настройке его параметров перед запуском.
  - Пример: **docker create -p 8080:80 myapp:latest** создаст контейнер на основе образа **myapp:latest**, но не запустит его.

Итак, разница заключается в том, что **docker run** создает и сразу запускает контейнер, в то время как **docker create** только создает контейнер, который может быть запущен позже с использованием **docker start**.

24) Можно ли использовать JSON вместо YAML в файле для docker-compose, если да — как?

Да, так можно сделать. Для этого нужно явно указать имя файла, например так:

docker-compose -f docker-compose.json up

25) Расскажите о CMD и ENTRYPOINT в Dockerfile

Эти инструкции Dockerfile задают команду, исполняемую при запуске контейнера. При их использовании есть несколько правил, например:

- Должна быть минимум одна из них, **CMD** или **ENTRYPOINT**, в Dockerfile.
- Если контейнер используется как исполняемый файл — ENTRYPOINT должна быть определена.
- Если контейнер запускается с другими аргументами — CMD будет переопределена.

26) Как проверить версии Docker client и Docker server?

Версию Docker можно проверить с помощью **docker version [параметры]**. Если не указывать параметры, команда выдаст всю информацию, связанную с версией клиента и сервера. Чтобы получить только версию сервера, можно запустить такую команду:

docker version --format '{{.Server.Version}}'

27) Расскажите о процедуре входа в Docker Repository

Чтобы войти в Docker Repository, используется следующая команда:

docker login [OPTIONS] [SERVER]

Например, для входа в registry, размещенную локально, команда будет такой:

$ docker login localhost:8080

28) Расскажите о наиболее общих командах Docker

- **docker push**: Закачать репозиторий или образ в Registry;
- **docker run**: Запустить команду в новом контейнере;
- **docker pull**: Скачать репозиторий или образ из Registry;
- **docker start**: Запустить один или несколько контейнеров;
- **docker stop**: Остановить один или несколько контейнеров;
- **docker search**: Поиск образа на DockerHub;
- **docker commit**: Сохранить изменения в новый образ.

29) Опишите все возможные состояния контейнера Docker

Created — контейнер создан, но не активен.

Restarting — контейнер в процессе перезапуска.

Running — контейнер работает.

Paused — контейнер приостановлен.

Exited — контейнер закончил свою работу.

Dead — контейнер, который сервис попытался остановить, но не смог.

30) Где хранятся тома(volumes) Docker?

Тома(**volumes**), создаваемые и управляемые Docker хранятся в файловой системе сервера Docker по пути **/var/lib/docker/volumes/**. Тома — наиболее эффективный способ сохранения данных в Docker.