**1 вопрос:** Концепции виртуализации и контейнеризации

Начнём с определения виртуальная машина (ВМ) - компьютерная система, эмулирующая возможности каких-либо вычислительных комплексов гостевых платформ на аппаратно-программном обеспечении хост-плаформы. Виртуализация - создание среды, где приложения гарантированно работают, а влияние на ресурсы соседней виртуальной машины минимально.  
Визуализация делится на аппаратную и программную.  
Уровни виртуализации:

* операционные системы
* программное обеспечение
* аппаратные ресурсы
* рабочие столы
* сети
* данные
* системы хранения данных
* центры обработки данных

Контейнеризация - метод, с помощью которого программный код упаковывается в единый исполняемый файл вместе с библиотеками и зависимостями, чтобы обеспечить его корректный запуск.

Контейнер - формат пакетирования, который позволяет упаковать весь код и зависимости приложения в стандартный формат, чтобы приложение могло быстро и надёжно запускаться в разных вычислительных средах.  
По сути, контейнер универсален - его можно развернуть в любой операционной системе независимы от настроек.

Существуют 2 типа  конйтенейров:

* контейнеры приложений - применяются для организации микросервисов с горизонтальной масштабируемостью
* контейнеры операционной системы - применяются для хранения операционной системы и запуска нескольких процессов одновременно

Краткая история контейнеризации:

* 1963 - проект Project Mac от IBM и ОС Multics. Потом была ОС Control Program и Console Monitor System
* 1979 - выход 7-ой версии UNIX и появление системного вызова chroot (почти 20 лет был единственным способом контейнеризации), который заменял корневую директорую процесса и потомков на ту, что указана в параметрах вызоа
* 2001 - появление механизма Linux-VServer
* 2006 - Google разрабатывает Process Container
* 2007 - вызод Linux Container
* 2011 - реализация проекта Docker в инкубаторе Y Combination
* 2013 - компания Cloud Foundry создаёт Warden и выход Docker на рынок

Вирутальные машины недоутилизируют ресурсы сервера: процессор, память, диски выделяются на каждую ОС.

Контейнеры используют ресурсы и ядро хостовой ОС (они делают это оптимальнее).

Docker использует cgroups и namespaces:

Cgroups - технологии управления ресурсами системы и они состоят из:

* core - реализует иерархическую организацию процессов
* controllers - отвечают за распространение ресурсов системы CPU, memory и др.

Namespaces - механизм ядра Linux для распространения ресурсов состоят:

* пространство имён ползователей user namespace
* пространство имё индентификаторов PID namespace
* сетевой пространство имён network namespace
* пространство имён точек монтировнаия mount namespace
* пространство имён межпроцессного взаимодействия interprocess communication
* пространство имён UNIX Time-Sharing  
    
  **2 вопрос:** Развертывание и работа с объектами Kubernetes

Объекты - сущности, которые в архитектуре Kubernets используются для представления состояния кластера:

* Узлы
* Модули контейнеров или Поды
* Контроллеры
* Сервисы
* Тома (типы: emptyDir, hostPath, nfs, cephfs)
* Пространство имён

ConfigMap - это объект, позволяющий хранить и управлять конфигурационными данным приложения без изменения образов контейнеров. Применяется для хранения файлов конфигурации, переменных среды и других конфигурационных данных.

Secret -  это объект, используемый для хранения чувствительных даннхы, таких как пароли, ключи API или сертификаты.

Ingress - это объект, обеспечивающий доступ к вашим приложенияз из внешних сетей. Он контролируется внешние HTTP и HTTPS маршруты для приложений, работающих на кластере, и может преоставлять дополнительные функции (SSL/TLS-терминирование, балансировка нагрузки и аутентификация).

Persistent Volume - это объект, представляющий собой абстракцию от физического хранилища данных в кластере Kubernets. Независим от подключаемых хранилищ и определяет ресурсы хранения без деталей их использования.

Persistent Volume Claim - то заявка на использование Persisntent Volume, созданная разработчиком преложения.

Horizontal Pod Autoscale - это объект, автоматически масштабирующий количество копий модулей Pod на основе наблюдаемой загрузки CPU или пользовательских метрик.

Resource Quota - это объект, определяющий ограничения на количество ресурсов, которые могут быть использованы в определённом пространстве имён.

RBAC - механзим управления доступом в Kubernetes на основе ролей.

Operators - расшерения Kubernetes, разработанные для автоматизации управления, машстрабирования и обновления составных приложений и служб.

Kubectl - утилита командной строки для работы с Kubernetes. Она позволяет управлять кластеров, создавать, обновлять и удалять различные ресурсы Kubernetes.

Kubectl предоставляет множество команд для работы с различными ресурсами Kubernetes, такими как Pods, Deployments, Services и др.

* kubectl get: получение информации о ресурсах
* kubectl create: создание ресурса из файла конфигурации
* kubectl apply: используется для создания или обновления ресурсов в класете. Команда принимает файл или каталог с конфигурационными файлами YAML или JSON

Манифест Kubernetes - файл конфигурации, который может включать один или несколько вложенных файлов конфигурации объектов.

Примеры команд:

* создать или обновить ресурсы, описанные в файле: kubectl apply -f my-file.yaml
* создат ьили обновить ресурсы, описанные во всех файлах в директории: kubectl apply -f my-files-directory/

Синтаксис файла манифеста:

* apiVersion - версия API Kubernetes, используемая для создания этого объекта
* kind - создаваемый тип объекта
* metadata - данные, которые помогают однозначно идентифицировать объект, включая строку имени, UID и необязательно пространство имён
* spec - состояние, настраиваемое для объекта