- Pod
  - 概念及其组成
    - pod是k8s调度和管理的最小单位，而非容器，但pod本质上是一个逻辑概念
    - 为什么需要pod
      - 解决处理成组调度问题 ⚙ 假设有容器A、B、C，每个容器至少需要1G内存，且三个容器间使用文件交换进行通信
        - 将3个容器调度至同一个节点，但似乎并没有成组打包调度的现成工具可以使用
        - 实现一套分布式的文件系统，将3个容器分布在不同的节点上
        - 将容器组合成pod，为pod定义3G内存需要，由k8s调度
      - 将多个容器加入至同一个Linux Namespace命名空间中
        - 既然容器A、B、C需要紧密的结合在一起，共同提供服务，那么组合的最好方式就是将其塞到同一个命名空间中
        - 虽然Docker提供了 docker run --net --volumes-from 使得B容器加入至A容器的实现，但是如此一来A必须优先于B容器启动，这样就形成了拓扑关系，而非平等关系
        - 所以在pod中，有一个内置的容器: k8s.gcr.io/pause，称为infra容器，该容器永远都是第一个被创建的容器，生成Linux Namespace，后续所有的容器均加入到该Namespace中
        - Infra相当轻量，占用资源极少，其目的就是为了创建Namespace ⚙ 这也是为什么pod中的容器拥有共同的IP地址的原因
        - *需注意，文件系统并不在容器间共享（容器的文件系统源自rootfs）*
    - pod由一个或者多个容器组成，这些容器共享同一个Linux Namespace命名空间
      - pod中的容器一定全部位于同一个节点，而不会跨节点分布
      - pod的中文释义为豌豆荚，豌豆荚里面会有一个或多个豌豆，就如同容器一样。在一个豌豆荚里面，所以里面的豌豆聊起天来也很方便
    - 应用应基于多层应用和基于扩缩容被分布到多个pod中，即pod应该包含紧密耦合的容器组，例如sidecar(包含日志收集器，数据处理器等)
  - pod的创建
    - k8s鼓励使用Object API的声明式创建资源，而非直接使用命令行，如同ansible的ad-hoc模式与playbook模式一样
    - YAML描述文件
      - metadata ⚙ 包括名称，命名空间，标签等信息
      - spec ⚙ 包含pod内容的实际说明，例如pod的容器，卷和其它数据
      - *几乎所有的k8s资源描述文件均包含这两部分内容*
    - 可通过 kubectl explain 来发现可能的API对象字段
      - kubectl explain pod
      - kubectl explain pod.metadata
      - *准备manifest的依赖手段*
    - 创建方式
      - 创建 ⚙ kubectl create -f pod.yaml
      - 更新 ⚙ kubectl replace -f new_pod.yaml
      - UpdateOrCreate ⚙ kubectl apply -f pod.yaml ⚙ 更方便的方式，不用管到底是创建还是更新，由k8s决定
    - 关于Pod中的spec与运行时的status ⚙ 源码地址: https://github.com/kubernetes/kubernetes/blob/master/staging/src/k8s.io/api/core/v1/types.go
  - 拆解Pod YAML配置文件
    - metadata
      - name ⚙ 当前Pod的名称
      - namespace ⚙ 当前Pod所属的k8s命名空间，默认为default，生产中该值必须指定
      - labels ⚙ 该Pod所要添加的标签，由多个key-value组成
    - spec
      - containers
        - image ⚙ 当前容器所要运行的镜像
        - command ⚙ 相当于Docker的entrypoint，当该值缺省时，使用镜像的entrypoint
        - args ⚙ 提供给entrypoint的参数，当该值缺省时，使用Docker镜像的CMD
        - imagePullPolicy ⚙ 镜像拉取策略
          - Always ⚙ 每次创建Pod时重新拉取镜像，当镜像名称version为 :latest 时，该策略为默认值
          - Never ⚙ Pod永远不会主动拉取镜像
          - IfNotPresent ⚙ 只有当宿主机没有该镜像时才会拉取，当镜像名称version为非 :latest时，该策略为默认值
        - name ⚙ 可认为是容器名称，用于DNS_LABEL，所以在Pod中，该值唯一
        - lifecycle ⚙ 指定容器状态发生变化时需执行的动作
          - 状态
            - postStart ⚙ 在容器创建后，立即执行指定的动作，在entrypoint启动后执行，但不能保证该动作与entrypoint执行完成的顺序
            - preStop ⚙ 若postStart执行失败，k8s会根据容器的重启策略来决定如何处理
          - Hook Handler实现 ⚙ 在容器正常结束之前执行，需要注意的是，当容器出现异常而崩溃时，该动作不会被执行(可读/存活探针失效属于正常退出)
            - Exec ⚙ 运行特定的命令或者是shell脚本
            - HttpGet ⚙ 向容器发送http请求
        - livenessProbe ⚙ 存活探针，即周期性的健康检查
          - exec ⚙ 采用shell命令实现健康检查
          - httpGet ⚙ 采用http请求实现健康检查
          - periodSeconds ⚙ 每次执行健康检查的间隔时间，默认为10s
          - timeoutSeconds ⚙ 每次执行健康检查的超时时间，默认为1s
          - initialDelaySeconds ⚙ 启动容器后进行首次健康检查的等待时间
          - 当健康检查失败后，kubelet则会直接kill掉该容器，并根据其重启策略进行相应处理
        - readinessProbe ⚙ 就绪探针，判断当前容器是否已准备好提供服务
          - 内容与livenessProbe基本相同
          - 就绪探针若失败，kubelet会自动将其从Service的Endpoint列表中隔离出去，待后续恢复后重新添加至Endpoint列表，不会kill掉容器
        - ports ⚙ 列出当前容器所暴露的端口，算是一种给开发人员使用的补充信息，不会产生实际影响
          - containerPort ⚙ 容器暴露的端口
          - protocol ⚙ 端口协议，如TCP，UDP等
        - resources ⚙ 定义当前容器所需要的资源下限与上限
          - requests
            - cpu ⚙ 单位为CPU个数，若申请0.5个CPU，则可配置为"500m"
            - memory ⚙ 单位为"Ki"，"Mi"，"Gi"，如"256Mi"
          - limits ⚙ 与requests完全相同，定义资源上限
        - volumeMounts
          - mountPath ⚙ 容器内挂载路径，即volume应挂载至容器的哪个目录下
          - name ⚙ k8s Volume(挂载卷)的名称，必须与定义的Volume名称相同
          - readOnly ⚙ true/false，挂载卷是否只读
      - initContainers ⚙ 优先初始化容器组，定义优先于containers中容器启动的容器
      - nodeSelector ⚙ 节点标签选择器，指定Pod所调度的节点
      - restartPolicy ⚙ 指定Pod内容器重启策略，包含Always(默认)，OnFailure，Never
      - tolerations ⚙ 当前Pod的容忍策略，master节点默认会被打上 node-role.kubernetes.io/master "污点"
        - effect ⚙ 该值由node的taint确定，包含NoSchedule，PreferNoSchedule，NoExecute.
        - key ⚙ taint的键
        - operator ⚙ 匹配规则
          - Exists ⚙ node中的taint-key只要包含该值即可
          - Equal ⚙ 需要完全相等
        - tolerationSeconds ⚙ Pod能够容忍包含"污点"的节点的时间
        - value ⚙ taint的值
    - status ⚙ Pod运行时产生的数据，是DEBUG时重要的数据来源
      - conditions ⚙ 记录当前Pod的服务状态
        - lastProbeTime ⚙ Pod状态最后一次被探测的时间戳，即探针最后运行时间
        - lastTransitionTime ⚙ Pod最后一次发生状态变更的时间戳
        - message ⚙ 最后一次状态变更的可读信息
        - reason ⚙ 最后一次状态变更的原因
        - type
          - PodScheduled ⚙ Pod被调度至某一个节点
          - Ready ⚙ Pod以处于就绪状态，可对外提供服务
          - Initialized ⚙ 所有的init containers均初始化完毕
          - ContainersReady ⚙ 所有的containers均初始化完毕
      - containerStatuses ⚙ restartCount ⚙ 当前容器重启次数，非常重要的指标
      - phase ⚙ 记录当前Pod的宏观状态，非常重要的指标 值为k describe pod xxx 结果中的Status
        - Pending ⚙ Pod 的 YAML 文件已经提交给了 Kubernetes，API 对象已经被创建并保存在 Etcd 当中。但是，这个 Pod 里有些容器因为某种原因而不能被顺利创建。比如，调度不成功，镜像未下载完毕等
        - Running ⚙ Pod 已经调度成功，跟一个具体的节点绑定。它包含的容器都已经创建成功，并且至少有一个正在运行中，不能保证Pod是完全是正常运行的
        - Succeeded ⚙ Pod 里的所有容器都正常运行完毕，并且已经退出了。这种情况在运行一次性任务时最为常见
        - Failed ⚙ Pod 里至少有一个容器以不正常的状态（非 0 的返回码）退出。这个状态的出现，意味着你得想办法 Debug 这个容器的应用，比如查看 Pod 的 Events 和日志。
        - Unknown ⚙ 异常状态，意味着 Pod 的状态不能持续地被 kubelet 汇报给 kube-apiserver，这很有可能是主从节点（Master 和 Kubelet）间的通信出现了问题
  - 利用标签组织资源
    - pod
      - 一个pod可拥有多个标签，在不同的维度上对pod进行分类
      - 标签属于pod属性，而非容器属性，故定义在metadata.labels下
      - 使用 kebectl get po -l tag_name=tag_value 筛选出特定标签的pod
    - node
      - 对于集群中的节点，同样可以添加标签，目的在于更方便的调度 ⚙ 使用 kubectl label node node_name tag=tage_value 的方式为node添加标签
      - 在创建pod时，也可以指定node的标签，从而让pod调度至具有该标签的节点组中 ⚙ 定义在spec.nodeSelector中
    - namespace
      - 标签用于将node或者pod等资源进行分类，而namespace则提供逻辑隔离，如同Redis中的db0，db1一样
      - k8s namespace与Linux Namespace不同，前者用于将资源通过一种方式进行隔离，后者则是进程隔离工具
      - 创建namespace ⚙ kubectl create namespace namespace_name
      - 将pod加入至某一个namespace中 ⚙ 只需在 metadata.namespace中执行名称即可
  - volumes
    - K8s的卷是Pod的一个组成部分，并不是单独的k8s对象，不能单独创建与删除。此外，卷不属于容器的组成部分，容器只是引用或者挂载卷而已，所以不在containers中定义
    - 由于Pod中的每个容器都有自己独立的文件系统(来源于容器镜像)，所以为了达到容器间文件交换或者保存的目的，需要使用卷的方式完成
    - 可用卷类型
      - emptyDir ⚙ 用于存储临时数据的简单空目录，其生命周期与Pod生命周期相同。当Pod被销毁时，该目录同时也会被销毁，常用于Pod中容器的文件共享
        - field
          - medium ⚙ 指定用于emptyDir的存储介质，磁盘或内存
          - sizeLimit ⚙ 指定emptyDir的容量上限
      - hostPath ⚙ hostPath卷指向节点文件系统上的特殊文件或者目录，其生命周期不受Pod生命周期的影响，属于持久性存储
        - 绝大多数应用不应使用该类型的卷，因为Pod调度至哪个节点是不固定的，应用不应依赖具体的节点。因此，hostPath卷常被系统级别的Pod使用
      - configMap ⚙ k8s允许将配置选项分离到单独的资源对象ConfigMap中，本质上就是一个key-value键值对映射，值可以是简单的字面量，也可以是一个文件
        - ConfigMap的创建
          - 应用并不需要直接读取ConfigMap，映射的内容可通过环境变量或者卷文件的形式传递给容器
          - 命令行字面量创建 ⚙ kubectl create configmap configmap_name --from-literal=foo=bar --from-literal=key=value
          - 从配置文件中创建 ⚙ kubectl create configmap configmap_name --from-file=my.config
            - 创建key-value ⚙ JSON/YAML
            - 文件作为配置 ⚙ 文件内容可仅包含value值
            - 若配置文件中包含键值对，则创建键值对配置
            - 若配置文件仅包含值(value)，则key为文件名称，value为文件内容
          - 从配置文件目录中创建 ⚙ kubectl create configmap configmap_name --from-file=/path/to/dir
            - kubectl将遍历文件目录，以文件名作为key，文件内容作为value创建ConfigMap
        - ConfigMap向容器中传递
          - 作为环境变量注入 ⚙ 在containers.envFrom引用相关的ConfigMap即可
          - 作为卷挂载至容器 ⚙ 就像挂载emptyDir一样
            - 使用卷挂载完整条目ConfigMap的方式Pod可自动更新ConfigMap的修改 ⚙ 时效性有待实际测试
      - secret ⚙ 与ConfigMap基本类似，只不过属于加密卷，通常用户配置敏感信息，如数据库账户密码等
        - secret作为卷挂载至容器时，使用的存储介质为内存，而非磁盘