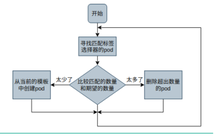
Controller

ReplicationController

- 基本概念：ReplicationController(简称rc)用于管理pod，确保pod始终处于运行状态，并且具备水平伸缩的功能
- 为什么需要rc
  - 确保一个或多个pod始终处于Running的状态，保证pod的持续运行：所以，使用rc管理的pod的restartPolicy必须是Always
  - 当集群节点发生故障时，rc可自动地将管理的pod在其余节点上创建副本
  - 方便地进行pod的水平伸缩，自动或手动均支持
- 组成
  - label selector：标签选择器，用于指定当前rc需管理哪些pod
    - 该值可省略，由k8s从pod模板中提取，使YAML文件更简洁
  - replica count：指定运行的pod数量
  - pod template：pod配置

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc-name
  namespace: rc-belong-ns
spec:
  replicas: number-of-pods
  selector:
    key: value
  template:
    pod-definition
```

- 控制循环
  - rc采用比较k8s集群中pod数量与期望值，对结果进行判断和处理
  - 这个过程是不断循环进行的，那么一旦实际pod数量与期望值不同，则rc会进行创建、删除等动作



  - 与事件驱动不同：类比select和epoll
- rc的水平缩放：kubectl scale rc rc-name --replicas=number
- 删除一个rc，但不删除所管理的pod：kubectl delete rc rc-name --cascade=false

ReplicaSet

- ReplicaSet(rs)和ReplicationController的行为完全相同，主要用于替换掉ReplicationController
- rs相比于rc，其pod选择器的表达能力更强。即支持多种方式标签的标签选择器，而不仅仅只是相等：单个rc无法将pod与标签env=prod与env=test同时匹配，只能匹配其中一个。而rs则可以匹配两组pod并将其视为一个大组
- rs.spec.selector
  - rs与rc YAML文件的最大区别就在于spec.selector，rc仅支持单个key-value的形式
  - matchLabels：rs最简单的key-value形式，支持多个key
  - matchExpressions
    - key：当前选择器要求该pod包含名为key的标签
    - operator：标签选择器的额外表达方式
      - In：标签的值必须与其中一个指定的values匹配
      - NotIn：标签的值与任何指定的values均不匹配
      - Exists：pod必须包含一个指定名称的标签(值不重要)。使用此运算符时，不应指定values字段
      - DoesNotExist：pod不得包含含有指定名称的标签。使用此运算符时，不应指定values字段
    - values：[]string：运算符的具体值列表

DaemonSet

- 特点
  - DaemonSet将pod部署到集群中所有的节点上，并且保证每个节点只有一个这样的pod在运行。
  - 当有新的节点加入 Kubernetes 集群后，该 Pod 会自动地在新节点上被创建出来；而当旧节点被删除后，它上面的 Pod 也相应地会被回收掉
  - 常用于部署系统监控应用程序，网络驱动程序等系统级别的应用
- DaemonSet将会绕过调度器，直接将pod调度到节点上
  - node通常会有一些taints(污点)，防止pod被部署到节点上。但是，诸如网络插件等应用，只有pod被部署到当前节点后，节点状态才会更新为Ready
  - 所以，DaemonSet将绕过调度器，直接将pod调度到节点上，使系统应用正常运行
- DaemonSet与Deployment一样，同样具有版本控制的能力，只不过Deployment是管理不同版本的ReplicaSet，而DaemonSet则是通过ControllerRevision对象进行版本控制的

Job

- Job资源允许用户定义只执行一次的任务，当pod内部进程成功结束后，不重启容器，一旦任务完成，pod就被认为处于完成状态
- 由于Job表示一次性运行的任务，故在定义pod模板时，其restartPolicy不能是默认的Always
  - OnFailure：pod在执行失败时重启
  - Never：pod永远不会重启，只会在失败时新建

重启次数或新建pod的数量由backoffLimit来确定

- 组成
  - activeDeadlineSeconds：执行Job的可运行期限，若超过时间任务仍未结束，则系统会尝试终止
  - backoffLimit：Job执行失败后的重试次数，默认为6
  - completions：指定Job需成功运行任务的次数，任务将以串行的方式成功运行指定的次数
  - parallelism：并行运行的Job pod数量
  - ttlSecondsAfterFinished：Job运行成功后的存活时间

假设completions为6，parallelism为4
则Job会首先创建4个任务运行，每完成一个任务，就会有新的任务创建出来。最终任务成功运行的次数等于completions

CronJob

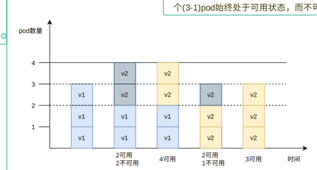
- k8s提供的定时任务支持，类似于Linux crontab
- 组成
  - jobTemplate：Job模板，而非Pod模板
  - schedule：定时计划，0 2 * * *，表示每天凌晨2点运行一次
- CronJob本质上是在计划时间内创建Job资源，而后由Job创建Pod执行实际的任务

Deployment

- 更新运行在pod内的应用程序
  - 当使用rs对pod进行管理控制时，不允许直接修改镜像，只能通过删除原有pod并使用新的镜像创建新的pod进行替换：即使使用kubectl edit对rs进行修改，镜像也不会同步更新至pod中
  - 更新方式
    - 直接删除所有pod，然后再创建新的pod
      - 会导致应用在一段时间内不可用
      - 首先修改rs中pod模板(修改pod模板并不会影响当前运行的pod)，而后一次性删除所有pod
    - 先创建新的pod，再删除旧的pod
      - 此时新旧版本pod共存，所以应用程序需进行版本兼容
      - 首先修改rs中pod模板(修改pod模板并不会影响当前运行的pod)，而后逐一删除pod
- Deployment资源
  - 为了更加方便、自动化地更新运行在pod中的应用，k8s提供了更加高阶的资源: Deployment，用于管理ReplicaSet
  - 在滚动升级的过程中，通常需要额外的引入另一个ReplicaSet，并协调两个Controller，使它们根据彼此不断修改，最终达到在不影响服务运行的情况下完成升级
  - deployment.spec
    - minReadySeconds
      - 指定新创建的pod至少要成功运行多久之后，才能将其视为可用，在pod可用之前，滚动升级不会继续。该值是滚动升级中非常重要的字段
      - 当pod中所有容器的就绪探针返回成功时，pod就会被标记为就绪状态，换而言之，pod从此刻开始成功运行。若在minReadySeconds内，就绪探针返回失败，则滚动更新将会停止

减缓升级速率，确保pod可用

    - paused：布尔值，表示当前升级是否暂停
    - progressDeadlineSeconds：升级操作的限定时间，默认为600s
    - replicas：由ReplicaSet所管理的pod的数量
    - revisionHistoryLimit：deployment允许回滚至前一个或多个版本，该值表示保留的历史版本数量，默认为10
    - strategy：更新策略
      - type
        - Recreate：该策略会一次性删除所有旧有pod，然后创建新的pod，通常用于应用程序无法进行版本兼容或不在乎应用一段时间不可用的情况
        - RollingUpdate：滚动升级，该策略会渐进地删除旧的pod，与此同时创建新的pod，使应用在整个升级过程中都处于可用状态，并确保其处理请求的能力没有因为升级而有所影响。该策略为默认值
      - rollingUpdate
        - 只有在更新策略为RollingUpdate时字段才有实际意义
        - maxSurge：决定了Deployment配置中期望的副本数以外，最多允许超出的pod实例的数量，既可以是百分位数(如%25)，也可以是pod的数量(如3)
          - 在滚动升级的过程中，pod的数量会超过replicas中定义的值，maxSurge则设定超出的上限，通常与服务器资源相关
        - maxUnavailable：决定了在滚动升级期间，相对于期望副本数能够允许有多少pod实例处于不可用状态，既可以是百分位数(如%25)，也可以是pod的数量(如3)
          - maxUnavailable是相对于期望副本数而言的。如果replica的数量为3，maxUnavailable的值为1，则整个更新过程中必须至少保持两个(3-1)pod始终处于可用状态，而不可用pod数量则可以超过1个



滚动升级时，maxSurge=1，maxUnabaiable=1的情况

  - 常用命令
    - 查看Deployment升级中的状态：kubectl rollout status deployment deployment-name
    - 暂停Deployment升级过程：kubectl rollout pause deployment deployment-name
    - 恢复Deployment升级过程：kubectl rollout resume deployment deployment-name
    - 取消Deployment升级过程或回滚至上一个版本：kubectl rollout undo deployment deployment-name：取消or回滚取决于Deployment的状态
    - 显示Deployment的升级历史：kubectl rollout history deployment deployment-name
    - 回滚到一个特定的Deployment版本：kubectl rollout undo deployment deployment-name --to-revision=specific-version