**原创；微信公众号：千里行走；**

**受限图片大小限制，有些图片不是很清晰，可以到微信公众号查看；**

**目录**

**(1).yaml方式容器化**

**(2).yaml关键配置**

**1.zookeeper核心端口**

**2.PodDisruptionBudget**

**3.PodAntiAffinity(pod反亲和性)**

**4.readiness(就绪探针)/liveness(存活探针)**

**5.权限声明**

**(3).生产环境注意事项**

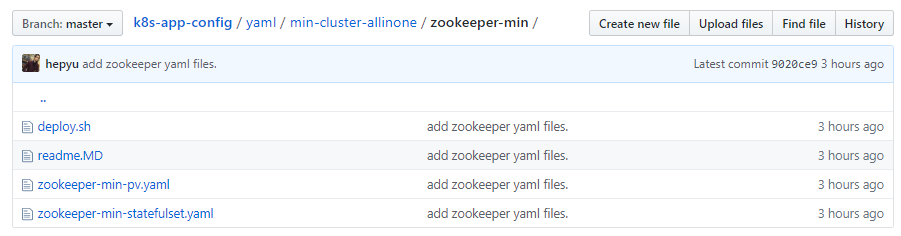
**(4).参考资料**

**正文**

**(1).yaml方式容器化**

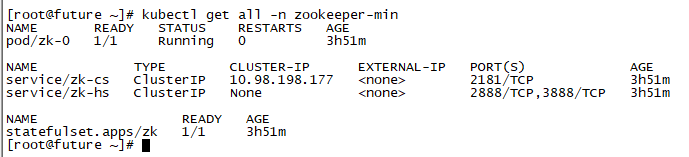
同样，与系列文章类似，笔者也提供了zookeeper容器化的配置文件，位于github：

<https://github.com/hepyu/k8s-app-config/tree/master/yaml/min-cluster-allinone/zookeeper-min>



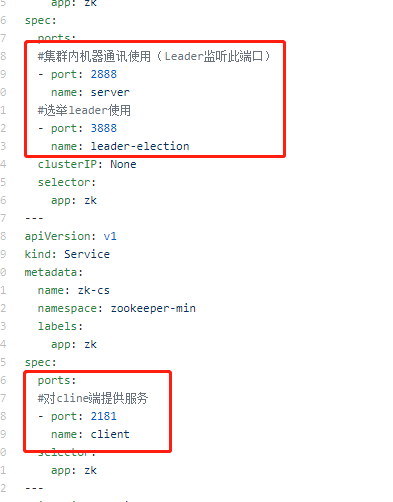
提供了shell部署脚本，直接执行：sh deploy.sh

查看组件：kubectl get all -n zookeeper-min



**(2).yaml关键配置**

**1.zookeeper核心端口**

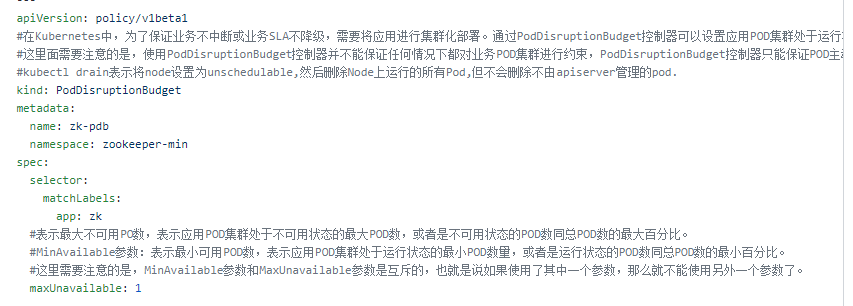


**2.PodDisruptionBudget**

在Kubernetes中，为了保证业务不中断或业务SLA不降级，需要将应用进行集群化部署。通过PodDisruptionBudget控制器可以设置应用POD集群处于运行状态最低个数，也可以设置应用POD集群处于运行状态的最低百分比，这样可以保证在主动销毁应用POD的时候，不会一次性销毁太多的应用POD，从而保证业务不中断或业务SLA不降级。

这里面需要注意的是，使用PodDisruptionBudget控制器并不能保证任何情况下都对业务POD集群进行约束，PodDisruptionBudget控制器只能保证POD主动逃离的情况下业务不中断或者业务SLA不降级，例如在执行kubectl drain命令时。

kubectl drain表示将node设置为unschedulable,然后删除Node上运行的所有Pod,但不会删除不由apiserver管理的pod。



**3.PodAntiAffinity(pod反亲和性)**

主要用于定义规则，一个worknode只能有一个zkPod，防止zk集群都集中部署在同一个wroknode而出现事实单点。

需要注意的是，在生产环境中，除了PodAntiAffinity之外，还要结合污点与nodePort，k8s只是用来运维zk的Matrix。



**4.readiness(就绪探针)/liveness(存活探针)**

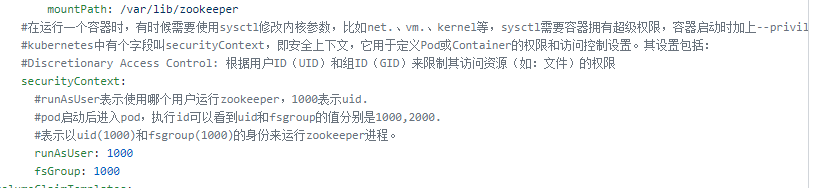
这个很好理解，就绪探针是告诉集群当前pod就绪，可以接受流量了；

存活探针是告诉集群当前POD是否还存活，如果dead需要重启pod；



**5.权限声明**

定义容器启动时，以什么样的用户身份/权限去启动进程。



**(3).生产环境注意事项**

**要注意三点：**

**1.使用PodAntiAffinity去定义同一个worknode不能出现同一个zk集群1个以上的zkPod，防止出现事实单点。**

**2.使用污点防止无关的pod被schedule到zk专用的worknode,这个worknode上到不一定只有zk，但是这个worknode是被规划的，不允许随意schedule pod。**

**3.使用nodePort暴露zk的2181端口给容器外部的服务使用。**

总的来说，我们不希望zkPod四处飘荡，是固定到有限个worknode，且有限个worknode的每一个worknode都有且只有同一个zk集群的一个zkPod，这样才具备高可用性，以及实用性，毕竟配置中心的zk地址是固定的，且当存在多个zk集群时不至于晕；K8S事实上是作为zk集群的Matrix存在；同样使用于其他中间件。

**(4).参考资料**

考资料：

1.Running ZooKeeper, A Distributed System Coordinator

https://kubernetes.io/docs/tutorials/stateful-application/zookeeper/

2.运行 ZooKeeper， 一个 CP 分布式系统

https://kubernetes.io/zh/docs/tutorials/stateful-application/zookeeper/

