# 云原生面试题

## docker面试题

1.什么是Docker？

Go语言开发，容器虚拟化技术，C/S架构，具有隔离、快速、轻便的特点。

2.Docker与虚拟机有何不同？

相比于虚拟机，docker更快速、轻便。  
快速：秒级启动  
轻便：联合文件系统

3.Docker镜像是什么？

联合文件系统。  
可以通过DockerFile构建或通过容器Commit生成。

4.Docker容器是什么？

镜像实例

5.Docker容器有几种状态？

四种状态：运行，暂停，重启，退出

6.DockerFile常见指令

FROM  
ADD COPY  
ENV WORKDIR  
RUN  
CMD ENTRYPOINT

7.Dockerfile中的命令COPY和ADD命令有什么区别？

COPY是复制文件  
ADD可以自动解压

8.Docker常用命令

容器、镜像、仓库、日志、进程、元数据、拷贝

9.容器与主机之间的数据拷贝命令

docker cp <主机路径> <容器ID>:<容器路径>  
docker cp <容器ID>:<容器路径> <主机路径>

10.启动nginx容器（随机端口映射），并挂载数据卷

docker run -d --name myNginx -P -v <主机路径>:<容器路径> nginx

11.什么是Docker Swarm？

与Docker引擎集成的集群管理

12.如何批量清理临时镜像文件？

docker rmi -f $(docker images -q)

13.如何查看镜像支持的环境变量？

docker run <镜像ID> env

14.本地的镜像文件都存放在哪里?

镜像文件路径：/var/lib/docker/image  
容器文件路径：  
/var/lib/docker/containers

15.构建Docker镜像应该遵循哪些原则?

1).尽量选取满足需求但较小的基础系统镜像  
2).清理编译生成文件、安装包的缓存等临时文件  
3).安装各个软件时候要指定准确的版本号，并避免引入不需要的依赖  
4).从安全的角度考虑，应用尽量使用系统的库和依赖  
5).使用Dockerfile创建镜像时候要添加.dockerignore文件或使用干净的工作目录

16.容器退出后，通过docker ps 命令查看不到，数据会丢失么？

docker ps -a

17.如何停止所有正在运行的容器?

docker kill $(docker ps -q)

18.如何退出容器不终止容器

Ctrl+P+Q

19.如何清理批量后台停止的容器

docker rm $(docker ps -aq)

20.如何查看容器日志？

docker logs <容器ID>

21.可以在一个容器中同时运行多个应用进程吗？

一般不推荐在同一个容器内运行多个应用进程

22.如何控制容器占用系统资源（CPU，内存）的份额？

docker run 时，可以  
使用-c|–cpu-shares[=0]参数来调整同期使用CPU的权重，  
使用-m|–memory参数来调整容器使用内存的大小。

23.Docker能在非Linux平台（Windows+MacOS）上运行吗？

可以

24.如何将一台宿主机的docker环境迁移到另外一台宿主机？

停止Docker服务，将整个docker存储文件复制到另外一台宿主机上，然后调整另外一台宿主机的配置即可

25.什么是docker-compose?

实现批量docker容器编排

## k8s基础面试题

1. 什么是Kubernetes？它的主要目标是什么？

Kubernetes是一个开源容器编排平台，用于自动化部署、扩展和管理容器化应用程序。它的主要目标是简化容器化应用的部署和管理，并提供弹性、可靠的应用程序编排。

2. 什么是Pod？

Pod是Kubernetes的最小调度和部署单元。它是一个包含一个或多个容器的逻辑主机，这些容器共享网络和存储资源，并且在同一主机上共享生命周期。

3. 什么是ReplicaSet？

ReplicaSet是Kubernetes的控制器之一，用于确保在集群中运行指定数量的Pod副本。如果Pod的数量少于指定的副本数，ReplicaSet将创建新的Pod副本；如果Pod的数量多于指定的副本数，ReplicaSet将删除多余的Pod。

4. 什么是Deployment？

Deployment是Kubernetes的控制器之一，用于声明性地管理Pod副本集。它允许定义Pod模板、副本数和更新策略，使得应用程序的部署和更新变得简单可控。

5. 什么是Service？

Service是Kubernetes的抽象层，用于暴露应用程序的一组Pod。它为这些Pod提供稳定的网络终结点，并允许它们通过服务发现进行通信。

6. 什么是命名空间（Namespace）？

命名空间是一种在Kubernetes集群中创建多个虚拟集群的机制。它可以用于隔离和管理不同的应用程序、团队或环境。

7. 如何进行应用程序的水平扩展？

可以使用Deployment的副本数字段来进行水平扩展。通过增加副本数，Kubernetes会创建更多的Pod副本以应对负载增加。

8. 如何在Kubernetes中进行滚动更新（Rolling Update）？

可以通过更新Deployment的Pod模板来进行滚动更新。Kubernetes会逐步替换现有的Pod副本，确保在整个更新过程中应用程序的可用性。

9. 如何在Kubernetes中进行滚动回滚（Rollback）？

可以使用Deployment的回滚功能来进行滚动回滚。通过指定回滚到的特定修订版本或回滚到上一个修订版本，Kubernetes会自动恢复旧的Pod副本。

10. 什么是Kubernetes的水平自动扩展（Horizontal Pod Autoscaling）？

水平自动扩展是Kubernetes的功能之一，根据应用程序的负载自动调整Pod副本数。它基于CPU利用率或自定义指标来进行自动扩展。

11. 如何进行存储卷（Volume）的使用？

可以使用存储卷将持久化数据附加到Pod中。Kubernetes支持多种类型的存储卷，如空白存储卷、主机路径、持久卷等。

12. 什么是ConfigMap和Secret？

ConfigMap用于存储应用程序的配置数据，而Secret用于存储敏感信息，如密码、API密钥等。它们可以作为环境变量、命令行参数或挂载到容器中使用。

13. 什么是亲和性（Affinity）和反亲和性（Anti-Affinity）？

亲和性和反亲和性是Pod调度的约束条件。通过使用亲和性，可以将Pod调度到指定的节点；通过使用反亲和性，可以避免将Pod调度到指定的节点。

14. 什么是DaemonSet？

DaemonSet是Kubernetes的控制器之一，用于在集群中的每个节点上运行一个Pod副本。它适用于在集群中的每个节点上运行系统级别的守护进程。

15. 什么是Ingress？

Ingress是Kubernetes的资源之一，用于将外部流量路由到集群内的服务。它可以提供负载均衡、SSL终止、路径基于的路由等功能。

16. 什么是持久卷（Persistent Volume）和持久卷声明（Persistent Volume Claim）？

持久卷是一种Kubernetes资源，用于提供独立于Pod的持久化存储。持久卷声明用于请求持久卷，使得Pod可以访问持久化存储。

17. 什么是Init容器（Init Container）？

Init容器是Pod中的一个额外容器，用于在主应用程序容器启动之前运行初始化任务。它可以用于数据准备、配置下载等任务。

18. 如何在Kubernetes中进行配置文件的安全管理？

可以使用Secret来安全地管理敏感配置信息，如数据库密码、API密钥等。可以通过加密、访问控制和密钥轮换等措施来确保Secret的安全性。

19. 如何监控Kubernetes集群？

可以使用Kubernetes内置的指标和日志系统，如kube-state-metrics、Heapster和EFK堆栈，来监控集群的运行状态和性能。

20. 如何进行跨集群部署和管理？

可以使用Kubernetes Federation或Kubernetes多集群（Multi-cluster）解决方案来进行跨集群部署和管理。

21. 什么是Kubernetes的生命周期钩子（Lifecycle Hook）？

生命周期钩子是Pod中的回调函数，可以在容器的生命周期事件发生时触发。它们可以用于在容器启动、停止或失败时执行定制化操作。

22. 什么是Pod的探针（Probe）？

Pod的探针用于定期检查容器的健康状态。Kubernetes支持三种类型的探针：存活探针（Liveness Probe）、就绪探针（Readiness Probe）和启动探针（Startup Probe）。

23. 什么是Kubernetes的安全性措施？

Kubernetes提供了多种安全性措施，如访问控制、网络策略、身份验证和授权、安全上下文等。此外，还可以使用第三方工具和插件来增强Kubernetes的安全性。

24. 什么是容器资源限制（Resource Limit）和容器资源请求（Resource Request）？

容器资源限制用于限制容器使用的CPU和内存资源。容器资源请求用于向调度器声明容器所需的CPU和内存资源。

25. 什么是Kubernetes中的水平和垂直扩展？

水平扩展（Horizontal Scaling）指的是增加Pod副本数来处理更多的负载。垂直扩展（Vertical Scaling）指的是增加或减少单个Pod的资源限制。

26. 什么是Kubernetes的节点亲和性（Node Affinity）？

节点亲和性用于将Pod调度到具有特定标签或节点选择器匹配的节点上。它可以用于确保Pod运行在特定类型的节点上，如SSD存储节点或GPU节点。

27. 什么是Kubernetes的事件（Event）？

事件是Kubernetes集群中发生的重要操作或状态更改的记录。可以使用kubectl命令或API查看集群中的事件。

28. 什么是Helm？

Helm是Kubernetes的包管理工具，用于简化应用程序的部署和管理。它允许定义和版本化应用程序的Charts（图表），并通过Helm命令进行安装、升级和删除。

29. 如何进行Kubernetes集群的高可用性配置？

可以使用Kubernetes的Master节点高可用性（HA）模式，通过配置多个Master节点实现集群的高可用性。这可以通过使用负载均衡器、备份ETCD数据存储等方法来实现。

30. 什么是Kubernetes的状态管理器（StatefulSet）？

StatefulSet是Kubernetes的控制器之一，用于管理有状态应用程序的部署。它确保Pod的稳定网络标识和有序启动、停止，适用于数据库、队列等有状态应用程序。

31. 什么是Kubernetes的自定义资源定义（Custom Resource Definition，CRD）？

CRD允许用户将自定义资源（Custom Resources）引入到Kubernetes中。这使得用户可以扩展Kubernetes的API和控制器，以支持自定义的资源类型。

32. 什么是Kubernetes的配置管理工具？

Kubernetes提供了多种配置管理工具，如kubectl、kubeconfig文件、ConfigMap、Secret、Helm等。这些工具可以用于管理和传递应用程序的配置信息。

33. 什么是Kubernetes的网络模型？

Kubernetes的网络模型基于容器间和容器与外部的通信。每个Pod都具有唯一的IP地址，并且可以通过服务和Ingress来实现内部和外部的网络通信。

34. 什么是Kubernetes的升级策略？

Kubernetes的升级策略指定了如何处理应用程序的升级。它包括滚动更新、蓝绿部署、金丝雀发布等不同的升级方式。

35. 什么是Kubernetes的监控和日志记录解决方案？

Kubernetes提供了多种监控和日志记录解决方案，如Prometheus、Grafana、ELK堆栈等。这些工具可以用于监控集群的性能指标和应用程序日志。

36. 请解释一下 Kubernetes 的主要组件。

Kubernetes的主要组件包括：Master组件（API Server、Controller Manager、Scheduler）和Node组件（kubelet、kube-proxy、容器运行时）。

37. 如何在 Kubernetes 中扩展应用程序？

可以使用ReplicaSet、Deployment或Horizontal Pod Autoscaler（HPA）来扩展应用程序。

38. 怎样从一个镜像创建一个 Pod？

可以使用kubectl命令行工具或编写一个Pod的YAML文件，然后使用kubectl apply命令创建Pod。

39. 如何将应用程序部署到 Kubernetes？

可以使用Deployment、StatefulSet或DaemonSet等资源对象来部署应用程序。

40. 如何水平扩展 Deployment？

可以通过更改Deployment的副本数来水平扩展应用程序。例如，使用kubectl scale命令或更改Deployment的replicas字段。

41. 怎样在 Kubernetes 中进行服务发现？

可以使用Kubernetes的Service对象来进行服务发现。Pod可以通过Service的DNS名称进行通信。

42. 如何进行滚动更新（Rolling Update）？

可以通过更新Deployment的Pod模板或修改RollingUpdate策略来执行滚动更新。

43. 什么是 PVC（Persistent Volume Claim）？

PVC是用于声明对持久卷（Persistent Volume）的需求的对象，它允许Pod使用持久化存储。

44. 怎样进行容器间通信？

可以使用Pod的内部IP地址和端口号进行容器间通信。此外，也可以使用Service对象来提供稳定的网络访问。

45. 如何进行集群内部的日志收集？

可以使用Kubernetes的日志收集器（如Fluentd、Prometheus）来收集集群中各个Pod的日志。

46. 什么是亲和性调度（Affinity Scheduling）？

亲和性调度是一种机制，用于将Pod调度到特定的节点或一组节点，以便满足特定的调度策略。

47. 怎样进行水平自动扩展（Horizontal Autoscaling）？

可以使用Horizontal Pod Autoscaler（HPA）对象来自动根据CPU或其他指标水平扩展Pod副本的数量。

48. 如何进行安全访问控制（RBAC）？

可以使用Kubernetes的Role-Based Access Control（RBAC）机制来定义和管理用户对集群资源的访问权限。

49. 什么是 Sidecar 容器？

Sidecar容器是与主应用程序容器共同运行的辅助容器，用于提供额外的功能或服务。

50. 怎样进行跨集群部署？

可以使用Kubernetes的Federation机制来管理跨多个集群的应用程序部署和资源。

51. 如何进行存储卷的扩展和快照？

可以使用Kubernetes的存储类（StorageClass）和持久卷声明（Persistent Volume Claim）来管理存储卷的扩展和快照。

52. 什么是 Pod 的生命周期？

Pod的生命周期包括Pending、Running、Succeeded、Failed和Unknown等阶段。

53. 怎样进行热更新（Hot Deployment）？

可以使用滚动更新策略，将新版本的应用程序容器逐步替换旧版本的容器，实现热更新。

54. 什么是 Downward API？

Downward API是一种机制，用于将Pod的元数据（如标签、注解、环境变量等）注入到容器中。

55. 怎样进行资源限制和配额管理？

可以使用Kubernetes的资源限制（Resource Limit）和配额（Resource Quota）来控制Pod使用的资源。

56. 什么是 CNI（Container Network Interface）？

CNI是一种规范，用于定义容器运行时和网络插件之间的接口，以实现容器网络的配置和管理。

57. 如何进行多集群部署？

可以使用Kubernetes的多集群管理工具（如Kubefed）来管理和部署多个独立的Kubernetes集群。

## k8s升阶面试题

1. Docker和虚拟机有那些不同

虚拟化环境下每个 VM 是一台完整的计算机，在虚拟化硬件之上运行所有组件，包括其自己的操作系统。

容器之间可以共享操作系统,比起 VM 被认为是更轻量级.且与 VM 类似，每个容器都具有自己的文件系统、CPU、内存、进程空间.

容器与虚拟化比有以下优点:

* 敏捷应用程序的创建和部署：与使用 VM 镜像相比，提高了容器镜像创建的简便性和效率。
* 持续开发、集成和部署：通过快速简单的回滚（由于镜像不可变性）， 提供可靠且频繁的容器镜像构建和部署。
* 关注开发与运维的分离：在构建、发布时创建应用程序容器镜像，而不是在部署时， 从而将应用程序与基础架构分离。
* 可观察性：不仅可以显示 OS 级别的信息和指标，还可以显示应用程序的运行状况和其他指标信号。
* 跨开发、测试和生产的环境一致性：在笔记本计算机上也可以和在云中运行一样的应用程序。
* 跨云和操作系统发行版本的可移植性：可在 Ubuntu、RHEL、CoreOS、本地、 Google Kubernetes Engine 和其他任何地方运行。
* 以应用程序为中心的管理：提高抽象级别，从在虚拟硬件上运行 OS 到使用逻辑资源在 OS 上运行应用程序。
* 松散耦合、分布式、弹性、解放的微服务：应用程序被分解成较小的独立部分， 并且可以动态部署和管理 - 而不是在一台大型单机上整体运行。
* 资源隔离：可预测的应用程序性能。
* 资源利用：高效率和高密度。

2. 简述K8s和Docker关系

Docker是一个开源的应用容器引擎.通过Dockerfile中的配置,将应用,配置与依赖打包成一个镜像,通过镜像实现应用的部署.

K8s是开源的容器集群管理系统,可以实现容器集群的自动化部署,自动扩缩容,维护等功能.

K8s是一个可移植、可扩展的开源平台，用于管理容器化的工作负载和服务，可促进声明式配置和自动化。

K8s提供了以下容器不具备的功能:

* 服务发现和负载均衡
* 存储编排
* 自动部署和回滚
* 自动完成装箱计算
* 自我修复
* 密钥与配置管理
* 3. 简述Kube-proxy ipvs和iptables的异同
* ipvs和iptables都是基于Netfilter实现.
* iptables是为防火墙设计的它采用规则表实现,service数量越多iptables的规则条目越多,而iptables是从上到下逐一匹配,就会导致效率低下.
* ipvs采用hash表实现,当service数量达到一定规模时,hash查表的速度优势会显现出来,从而提高service的服务性能.

iptables的优点:

* 灵活
* 功能强大(根据tcp连接状态对包进行控制)
* 默认安装,老版本兼容
* ipvs的优点:
* 支持hash转发效率高
* 支持backend健康检测
* 调度算法丰富(静态算法:轮询rr,加强轮询wrr,目的地址dh,源地址sh,动态算法:最少连接lc,加权最少连接wlc,局部最少连接数LBLC,带复制的局部最少连接数LBLCR,最短延迟NQ,加权最少连接增强版SED…)
* 可以动态设置ipset集合

5. 简述微服务部署中的蓝绿发布

常见发布有:蓝绿发布,金丝雀发布,滚动发布

蓝绿发布主要用作测试环境,正式环境多使用金丝雀或滚动发布

蓝绿部署(Blue/Green Deployment)一共有2套系统.一套是正在提供服务的系统,标记为"绿色";另一套是准备发布的系统,标记为"蓝色".两套系统都是功能完善的,正在运行的系统,只是系统版本和对外提供服务的情况不同.

蓝色系统用来做发布前测试,测试发现问题直接在蓝色系统上进行修改,不干扰用户正在使用的绿色系统.

蓝色系统经过反复的测试,修改,验证,确定达到上线标准后,直接将用户切换到蓝色系统.

优点:

* 升级切换与回退速度快
* 版本切换时无需停机,风险最小

不足之处:

* 它是一个全量切换.无法切分流量
* 需要2倍的资源
* 如果蓝色版本有问题,对用户体验有影响

6. 什么是灰度发布

灰度发布:也被叫作金丝雀发布。灰度发布属于增量发布方法。也就是说，服务升级的过程中，新旧版本会同时为用户提供服务。

先将一小部分用户流量切换到新版本上,测试版本性能与功能.确认没有问题后在对整个集群进行升级.灰度发布就是把部署好的服务分批次,逐步暴露给越来越多的用户,最终完成版本切替上线.

优点

* 灵活,粒度细
* 无需(少量)侵入代码
* 用户无感知,平滑过渡

不足:

* 对自动化要求较高

部署过程:

* 从LB拆除灰度服务器,升级后再加入
* 少量用户切换到新版本进行测试
* 如果没问题,调整灰度服务器数量,重复步骤2将更多的用户流量切换到新版本.直至所有用户流量完成切换.

7. 什么是滚动发布

k8sdeployment是滚动发布.按批次依次替换老版本(maxSurge默认25%),逐步升级到新版本.发布过程中应用不中断,用户体验平滑.

8. 简述Kubernetes静态Pod

静态Pod由kubelet创建并且总是在kubelet所在节点上运行.

它们不能通过API Server进行管理,无法与==ReplicationController，Deployment,Daemonset==进行关联,它就是一个Pod,并且kubelet也无法对其健康检查

创建静态Pod有两种方式：配置文件方式 和HTTP方式。

配置文件方式: 通过在manifests目录下创建yaml,由kubelet定期扫描自动创建Pod资源

HTTP方式:kubelet启动参数–manifest-url,kubelet定期访问此url,下载Pod定义文件,以yaml或json格式进行解析,并创建Pod.

kubeadm安装集群配置文件路径:/var/lib/kubelet/config.yaml中staticPodPath: /etc/kubernetes/manifests

二进制安装配置文件路径:由/usr/lib/systemd/system/kubelet.service的–pod-manifest-path=值定义

etcd,kube-apiserver,kube-controller-manager,kube-scheduler都是静态Pod

如果需要创建自己的静态pod,直接将yaml放置在 /etc/kubernetes/manifests 目录下即可

配置文件方式创建:

/etc/kubernetes/manifests/static-web.yaml

文件内容如下

apiVersion: v1

kind: Pod

metadata:

name: static-web

labels:

role: myrole

spec:

containers:

- name: web

image: nginx

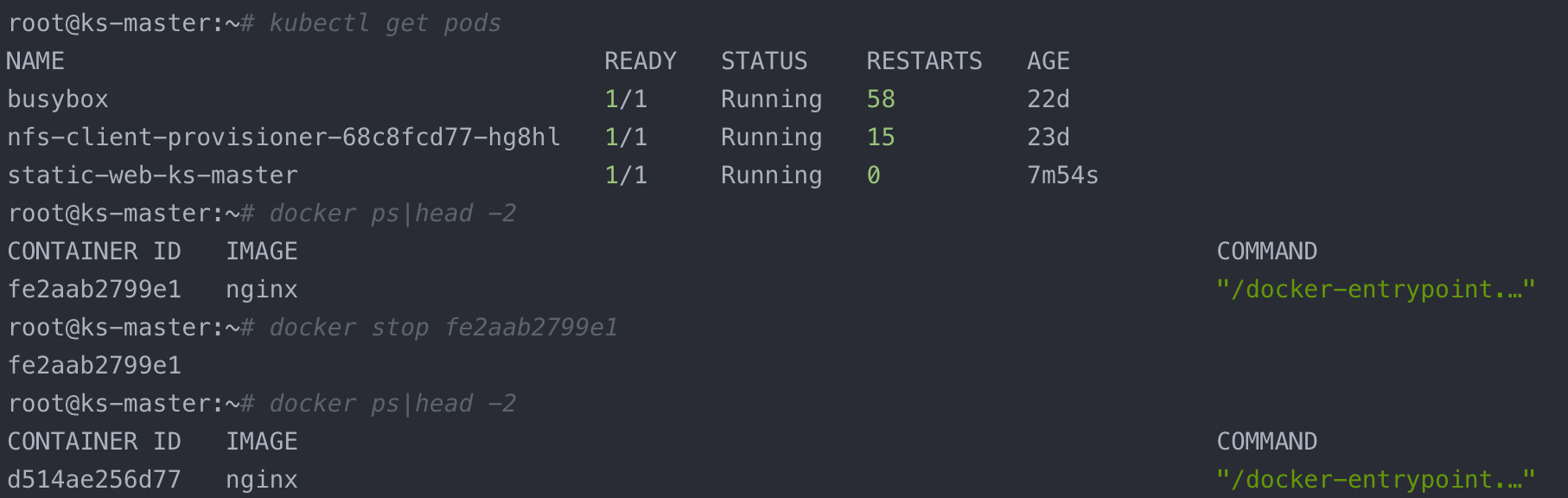
ports:

- name: web

containerPort: 80

protocol: TCP

当文件被创建后该pod就会被kubelet自动创建,且当停止该容器后马上会被自动再次创建.



想要删除把配置文件删除即可

9. 简述K8s存储卷有哪些,说明它们的特征和用途

* emptyDir: Pod调度到节点创建的临时空目录,Pod删除随之删除,用于容器存放临时文件
* hostPath: 节点存储卷,将宿主机的文件或目录挂载到Pod
* Pv/Pvc: 外部存储挂载到Pod,生命周期独立于Pod,具有持久性
* StorageClass: 充当PV模板,从而可以动态创建PV,自动构建相应的持久化数据存储
* configmap: 明文,热更新,支持多个pod共享,避免重复修改.
* secret: 密文

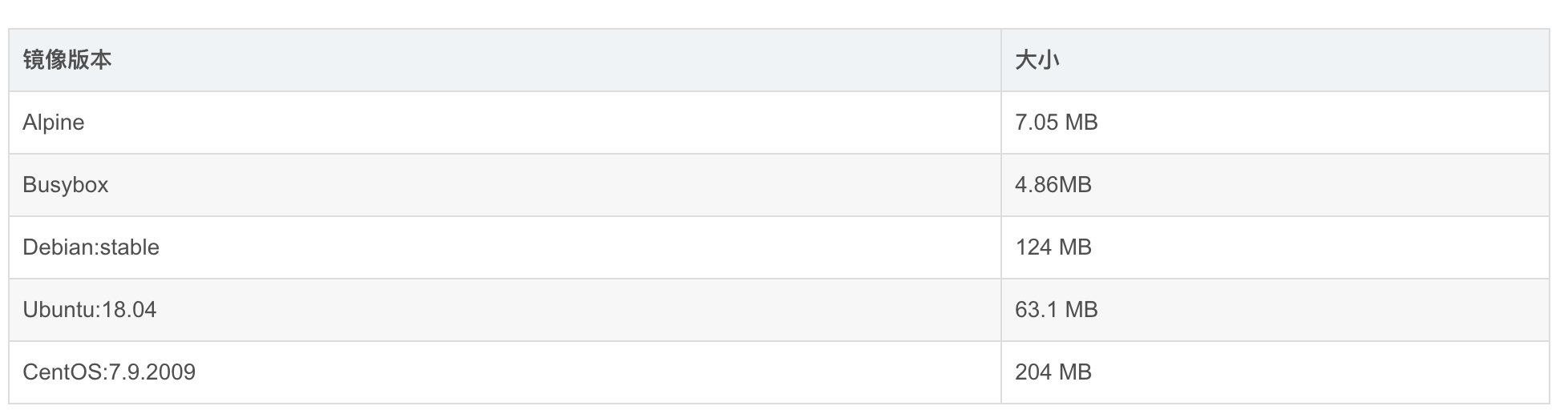
10. 简述Dockerfile中copy和add的异同

Copy: 将宿主机的文件复制到容器内

Add: 除了复制功能以外,还带有解压缩功能.支持url

copy后解压,需要先确认容器内有解压所需的命令.当对容器镜像大小要求比较苛刻时add显然更好.使用空镜像scratch时特别明显

常见较小的镜像有:



11. 简述Dockerfile中CMD 和ENDPOINTS区别

CMD: 指定启动容器时执行的命令，每个 Dockerfile只能有一条 CMD 命令。如果指定了多条命令，只有最后一条会被执行。如果用户启动容器时候指定了运行的命令，则会覆盖掉 CMD 指定的命令。

Endpoint:配置容器启动后执行的命令，并且不可被 docker run 提供的参数覆盖。每个 Dockerfile 中只能有一个 ENTRYPOINT，当指定多个时，只有最后一个起效。

两者区别:

1、CMD指令指定的容器启动时命令可以被docker run指定的命令覆盖ENTRYPOINT指令指定的命令不能被覆盖，而是将docker run指定的参数当做ENTRYPOINT指定命令的参数。

2、CMD与ENTRYPOINT同时存在时，CMD指令可以为ENTRYPOINT指令设置默认参数，而且CMD可以被docker run指定的参数覆盖；

13. 如何将SVC的NodePort设置成40000端口

默认svc NodePort范围是30000-32767

修改kube-apiserver.yaml添加–service-node-port-range=30000-45000

/etc/kubernetes/manifests/kube-apiserver.yaml

- --service-node-port-range=30000-45000

14. 简述K8s常用控制器及特点

* Deployment 无状态控制器,启动顺序是无序的,Pod名字和ip是随机的,后台存储往往是共享的.(web网站)
* StatefulSet 有状态控制器,有序启动,Pod名字固定,后台存储是独占的.(主从数据库)
* Daemonset 每个节点都部署.(日志采集和监控)
* CronJob 定时运行Pod
* Job 一次性运行Pod

15. K8s无法拉伸和收缩副本可能的原因

手动拉伸或者收缩

kubectl scale deployment nginx --replicas=2

生产中常用HPA实现Deployment的自动伸缩

命令行

## 设置deployment中容器的cpu和内存

kubectl set resources deployment nginx --limits=cpu=200m,memory=512Mi

## 设置autoscal

kubectl autoscale deployment nginx --min=4 --max=8 --cpu-percent=80

## 查看autoscal

kubectl get hpa

yaml方式

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: hpa-nginx

spec:

maxReplicas: 8

minReplicas: 4

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: nginx

targetCPUUtilizationPercentage: 80

当手动伸缩值与HPA的最小值冲突时,收缩不生效

当手动伸缩值与HPA的最大值冲突时,扩展不生效

当对deployment有requests参数时,当(namespace或节点)可用资源不满足时,扩展失效.

16. 简述Deployment控制器的升级和回滚

nginx-blue.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

revisionHistoryLimit: 5 # 记录多少个版本用来回滚

template:

metadata:

labels:

app: nginx

spec:

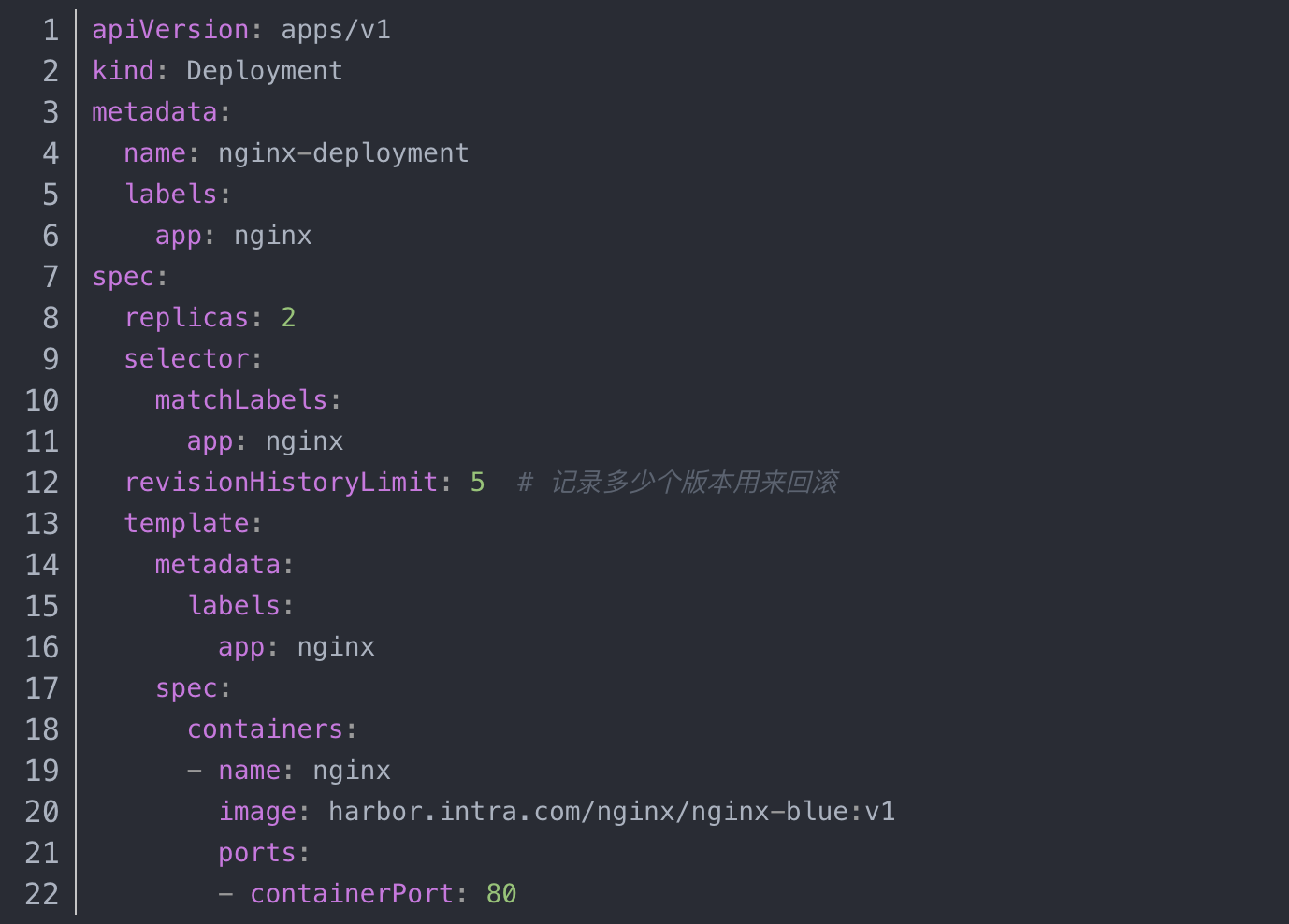
containers:

- name: nginx

image: harbor.intra.com/nginx/nginx-blue:v1

ports:

- containerPort: 80



部署生效

kubectl apply -f nginx-blue.yaml --record

访问测试

# curl 12.233.106.215

web-app1-blue version 2.0.0

部署green版本

cp nginx-blue.yaml nginx-green.yaml

sed -i 's/blue/green/g' nginx-green.yaml

kubectl apply -f nginx-green.yaml --record

# curl 12.233.106.223

web-app1-green version 1.0.0

查看版本

# kubectl rollout history deployment nginx-deployment

1 kubectl apply --filename=nginx-blue.yaml --record=true

2 kubectl apply --filename=nginx-green.yaml --record=true

回滚到blue版本

kubectl rollout undo deployment nginx-deployment --to-revision=1

# curl 12.233.106.224

web-app1-blue version 2.0.0

查看版本

# kubectl rollout history deployment nginx-deployment

2 kubectl apply --filename=nginx-green.yaml --record=true

3 kubectl apply --filename=nginx-blue.yaml --record=true

17. 简述Dockerfile中ENV和ARG的区别

相同:

​ 两个都是Dockerfile中的变量

​ 两者都不能被CMD和ENTRYPOINT

不同:

变量的生命周期

​ ARG的生命周期仅是docker构建中有效,一旦构建完ARG的值失效.

​ ENV的不仅构建时有效,当使用此镜像再次构建或者使用此镜像生成容器,ENV的值任有效.

变量值的传递参数

​ ARG构建时通过–build-arg参数将变量传给Dockerfile

​ ENV在docker run 命令中通过 -e标记来传递环境变量