# K8S技术及应用接入分享

资料整理
火龙果软件
荐

# 主要内容

- 整体架构
- 资源对象说明
- 部署过程与应用接入
- 常用命令
- 灰度发布

资料整理
火龙果软件
荐

# K8S简介

- Kubernetes的名字来自希腊，意思是“舵手”或“领航员”，K8S是将8个字母”ubernete”替换为8的缩写。
- Kubernetes是用于自动部署，扩展和管理容器化应用程序的开源系统，是Google一个开源项目，是Google10多年来大规模容器集群管理技术Borg的开源版本。
- Kubernetes是一个分布式容器集群管理系统，采用Go语言开发，是容器的上层技术架构。
- Kubernetes不局限于任何开发平台和开发框架，不限制支持应用的类型，不区分“app”和“services”，只要这个应用可以在容器里运行，那么就可以很好的运行在K8S。

资料整理
火龙果软件
荐

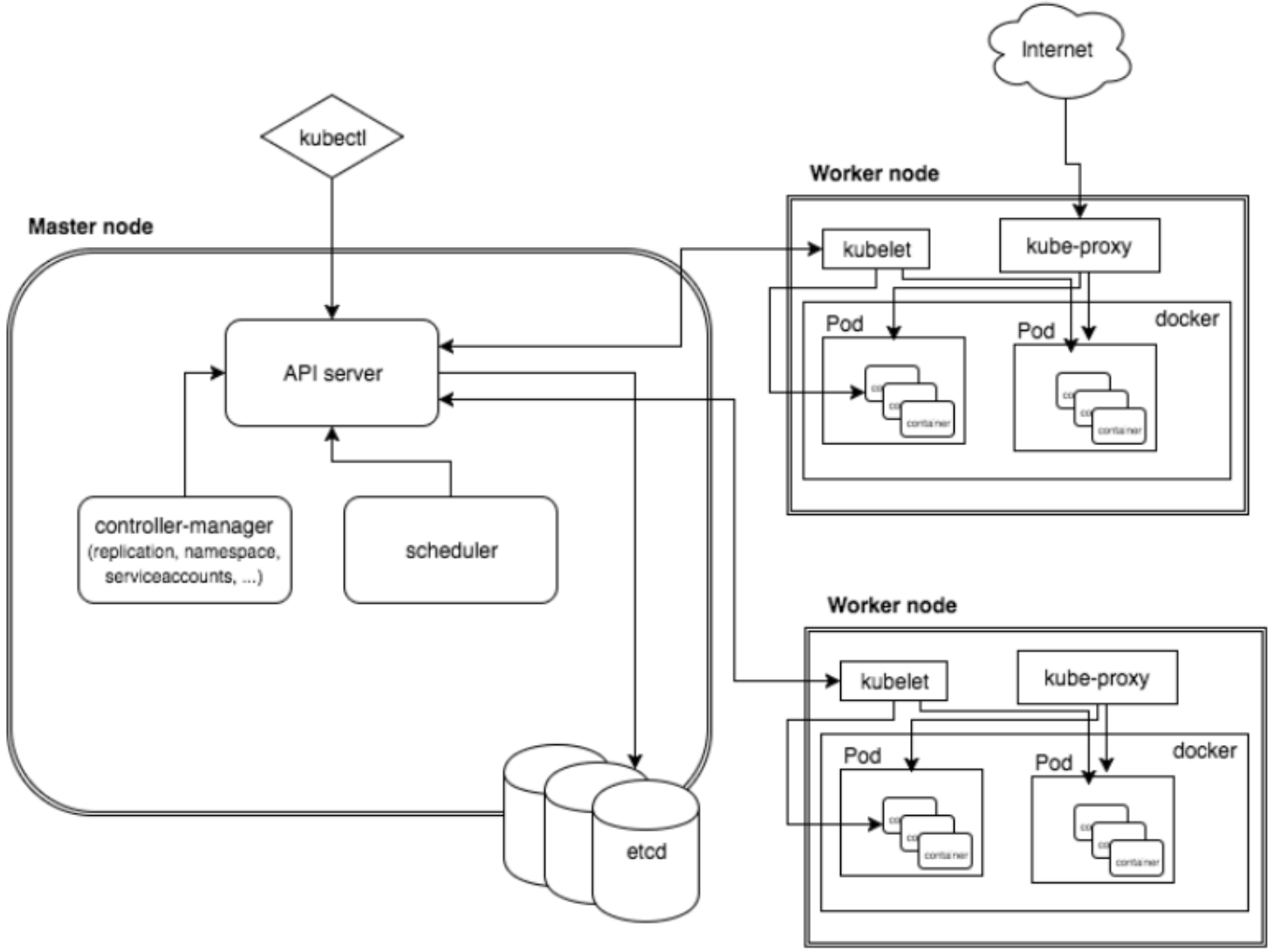
# K8S主要功能

- K8s能方便的管理容器集群应用，提供应用部署、维护、服务发现、负载均衡、故障自我修复、服务滚动升级、在线扩容，资源额度配置等多种功能。

- 全面支持微服务架构，可以随时的进行整体迁移，提供完善的管理工具，涵盖了开发、测试、运维的各个环节。

资料整理
火龙果软件
荐

# K8S集群整体架构




- Kubernetes属于主从分布式架构，主要由Master Node和Worker Node组成，以及包括客户端命令行工具kubectl和其它附加项如网络组件、路由转发、UI等。
- Master Node：作为控制节点，对集群进行调度管理；Master Node由API Server、Scheduler、Controller-Manger Server所组成。
- Worker Node：作为真正的工作节点，运行业务应用的容器；Worker Node包含kubelet、kube proxy和Container Runtime。
- Kubectl：用于通过命令行与API Server进行交互，而对Kubernetes进行操作，实现在集群中进行各种资源的增删改查等操作。
- Kubernetes默认使用etcd作为集群整体存储，etcd是一个简单的、分布式的、一致的key-value存储。

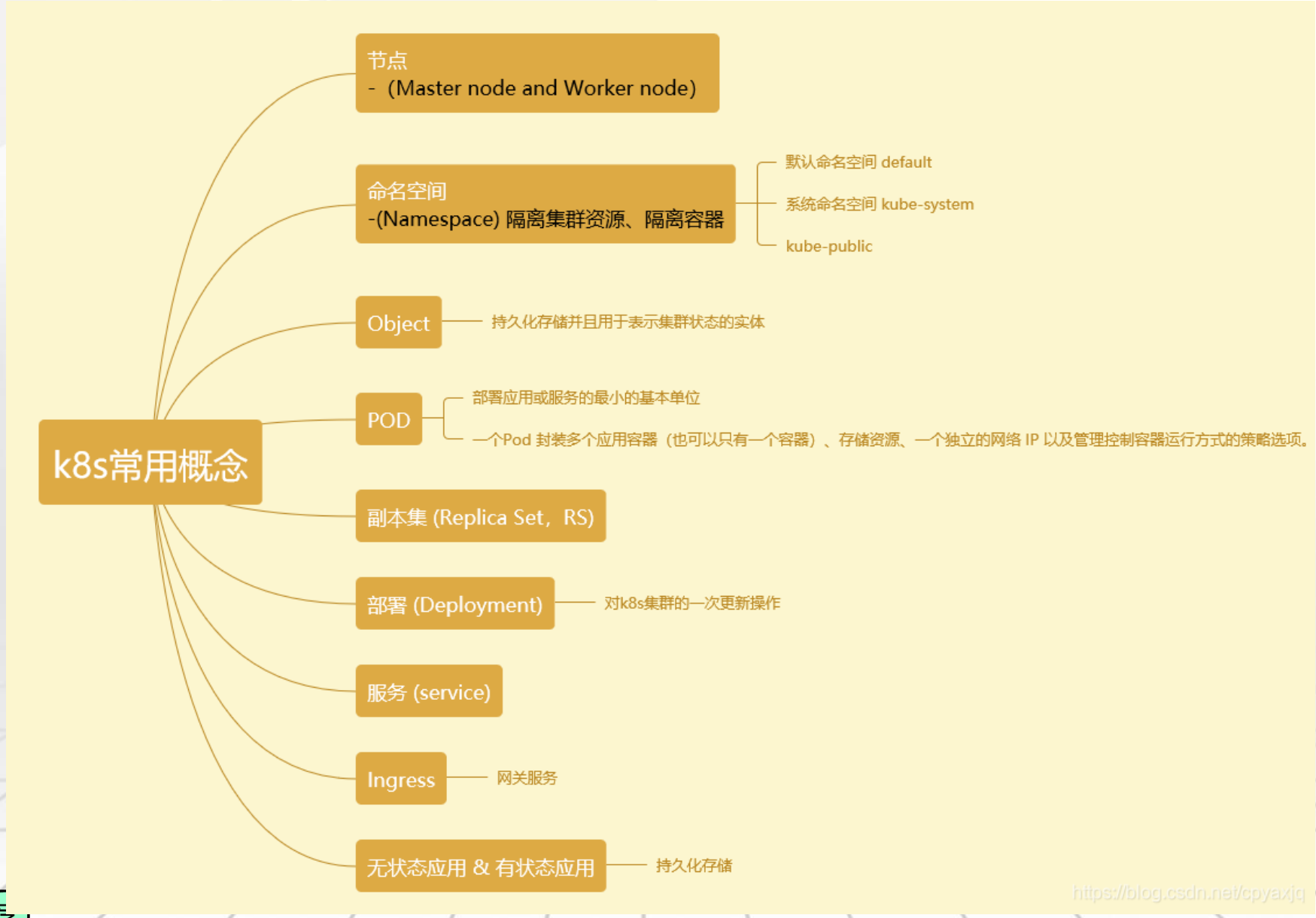
资料整理
火龙果软件
荐

# K8S集群架构组件

- Kube-apiserver 提供了HTTP Rest接口的关键服务进程，是k8s里所有资源的增删改查等操作的唯一入口，也是集群的入口进程，结果状态被持久存储于etcd中。
- Kube-scheduler k8s里负责所有资源调度的进程。
- Kube-controller-manager 集群内部的管理控制中心，负责容器编排，集群内的Node、Pod副本、命名空间、服务账号、资源限额等的管理。
- Kube-proxy 负责k8s中的网络代理，每个node上都运行一个kube-proxy守护进程，它能够按需为service资源对象生成iptables或ipvs规则，将service的流量转发至后端真正的pod中。
- kubelet 是运行在node上的守护进程， kubelet会在APIserver上注册当前节点，管理k8s中pod的生命周期，它从API server接收关于pod对象的配置信息，定期向Master汇报节点资源使用情况。

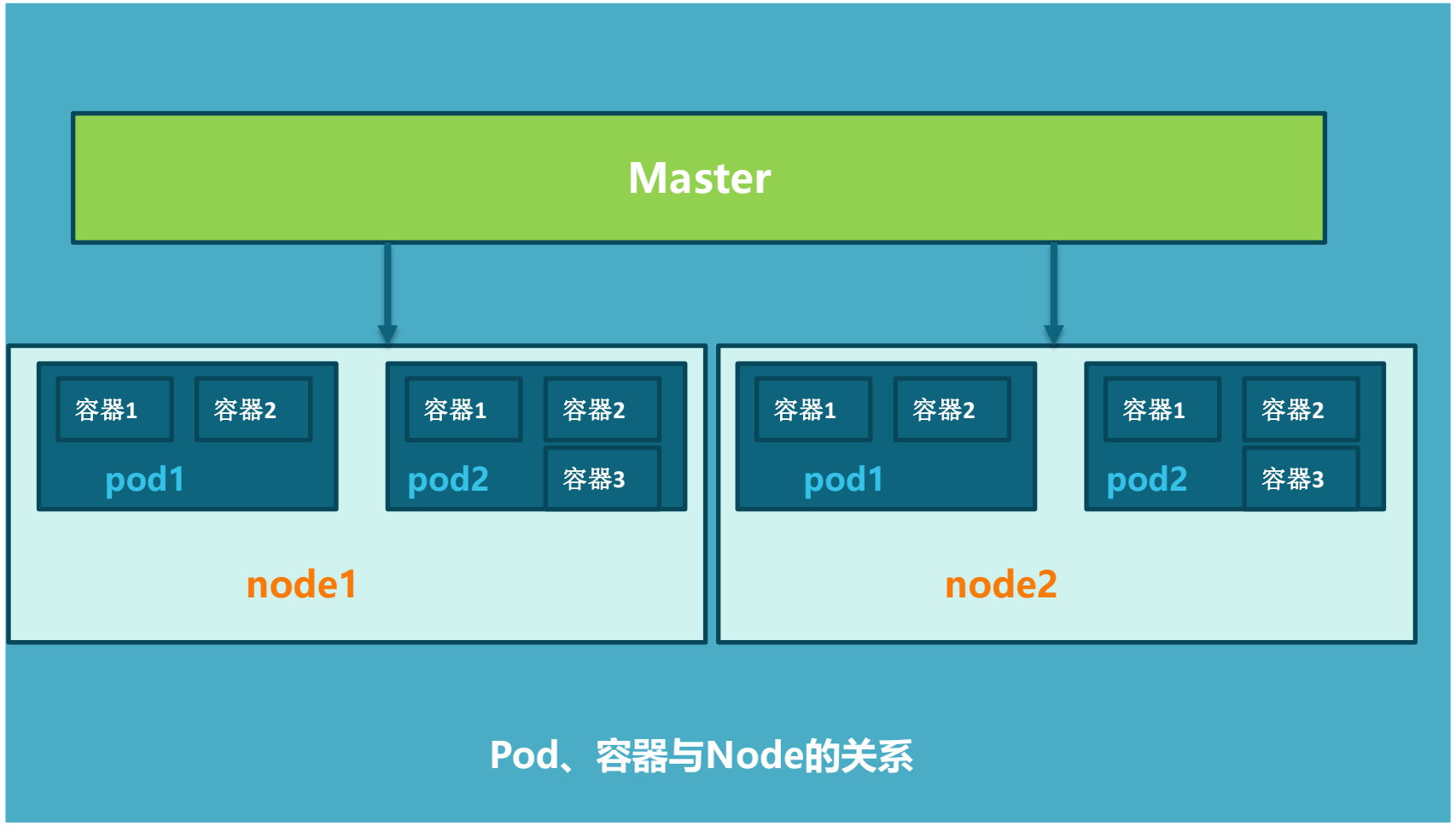
资料整理
火龙果软件
荐

# K8S常用概念




资料整理
火龙果软件

# 资源对象说明-Pod




- Pod是最小的部署单元，一个pod由一个或多个容器组成，pod中共享存储和网络。
- 如果pod所在的节点宕机，会将此节点的pod重新调度到其它节点上。
- docker volume对应k8s中的pod volume。
- 每个pod都可以设置限额的节点机CPU和内存资源。

资料整理
火龙果软件
荐

# 资源对象说明-deployment

- Deployment支持动态扩容、滚动升级、版本回滚，它用来生成对应的副本数，实现Pod副本的创建与更新过程。
- 在创建或删除pod时，实际需要先创建或删除deployment对象，然后再创建或删除pod，再由deployment来检查更新部署动作是否完成。

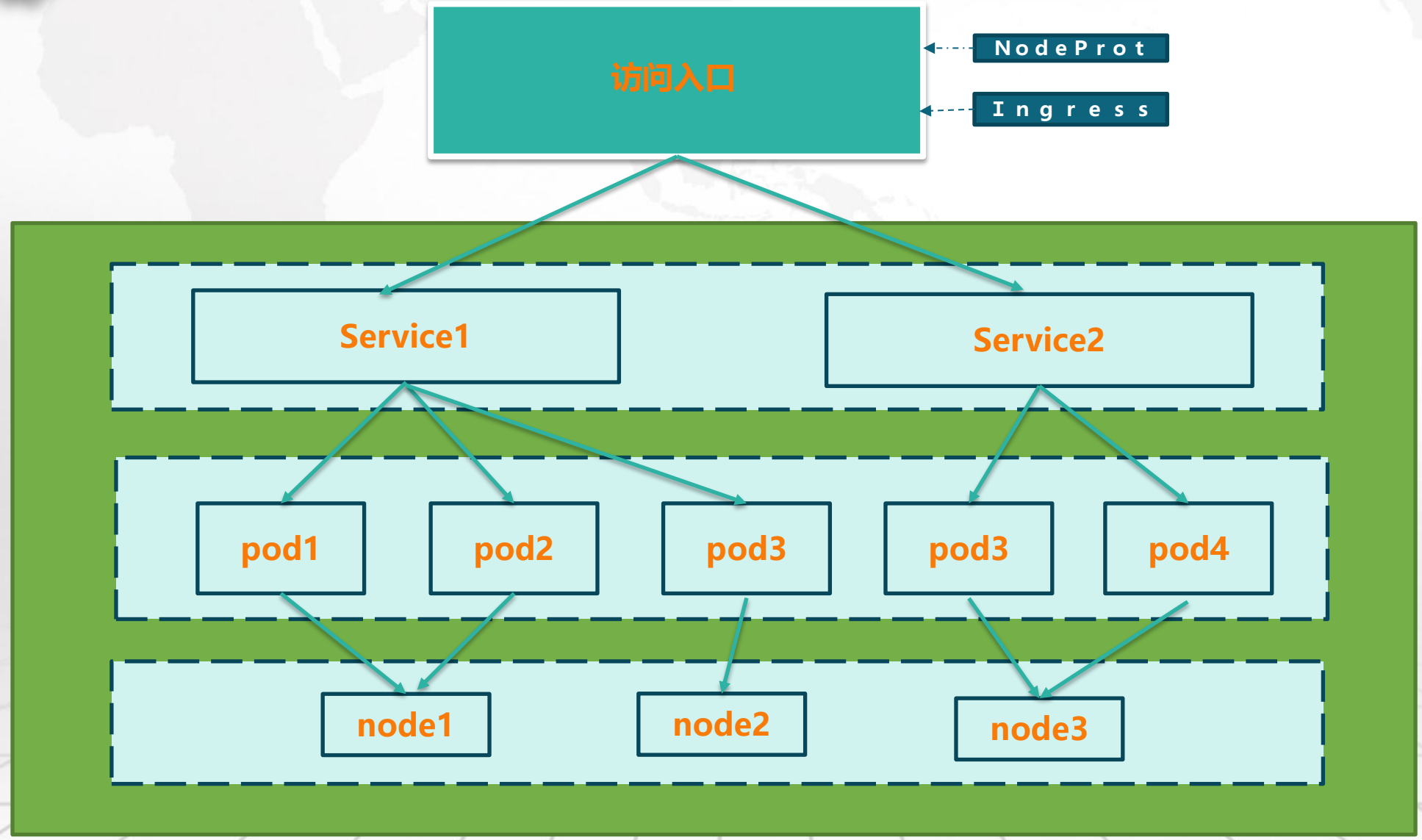
资料整理
火龙果软件
荐

# 资源对象说明-Service

- service是一个应用服务抽象，定义了访问pod集合的策略，它会被分配有一个Cluster IP， Kuber-proxy通过这个IP将请求转发到后端pod容器中，并且在pod内部实现负载均衡。
- service通过LableSelector来选择一组Pod提供服务。
- service提供三种对外访问类型，Ingress、NodePort、LoadBalancer。
- 每个pod都有一个PodIP，每个service都有一个ClusterIP，还有节点机自已的NodeIP，NodeIP、ClusterIP、PodIP在k8s集群中能自由访问，采用的是k8s自已设计的特殊路由规则，与我们常规的IP路由有很大的不同。
- 三种网络类型，node network，pod network ，service network，实现pod内的网络通信，pod和service之间的网络通信、service与外部的网络通信。

资料整理
火龙果软件
荐

# 资源对象说明-Service的访问




资料整理
火龙果软件
荐

# 资源对象说明-label

- lable可以附加到各种资源对象上，一个资源对象可以定义任意数量的lable。
- 给某个资源定义一个label，相当于给它打了一个标签，随后可以通过LableSelector标签选择器查询和筛选拥有lable的资源对象，以便于灵活、方便的进行资源分配、调度、配置、部署等管理工作。

资料整理
火龙果软件
荐

# 资源对象说明-Volume

- 数据卷，是Pod中能够被多个容器访问的共享目录，可以定义为和Pod生命周期相同的数据卷，也可以定义为映射到host宿主机上的共享存储路径中。

资料整理
火龙果软件
荐

# 资源对象说明-Ingress

- 通常情况下，service和pod只能在集群内部网络访问，而Ingress可以提供对外访问规则，并配置给service用于外部访问，简单的讲Ingree就是K8S应用的访问入口，在使用前需要先创建Ingress组件后再创建对应的应用Ingress资源。

资料整理
火龙果软件
荐

# 资源对象说明-namespace

- 命名空间namespace是一个所有对象资源的分组集合，各命名空间为软隔离状态，创建多个命名空间可以实现不同的环境，不同的使用用户，不同的项目来使用，如，开发环境、测试环境在同一个k8s集群的不同命名空间中使用。

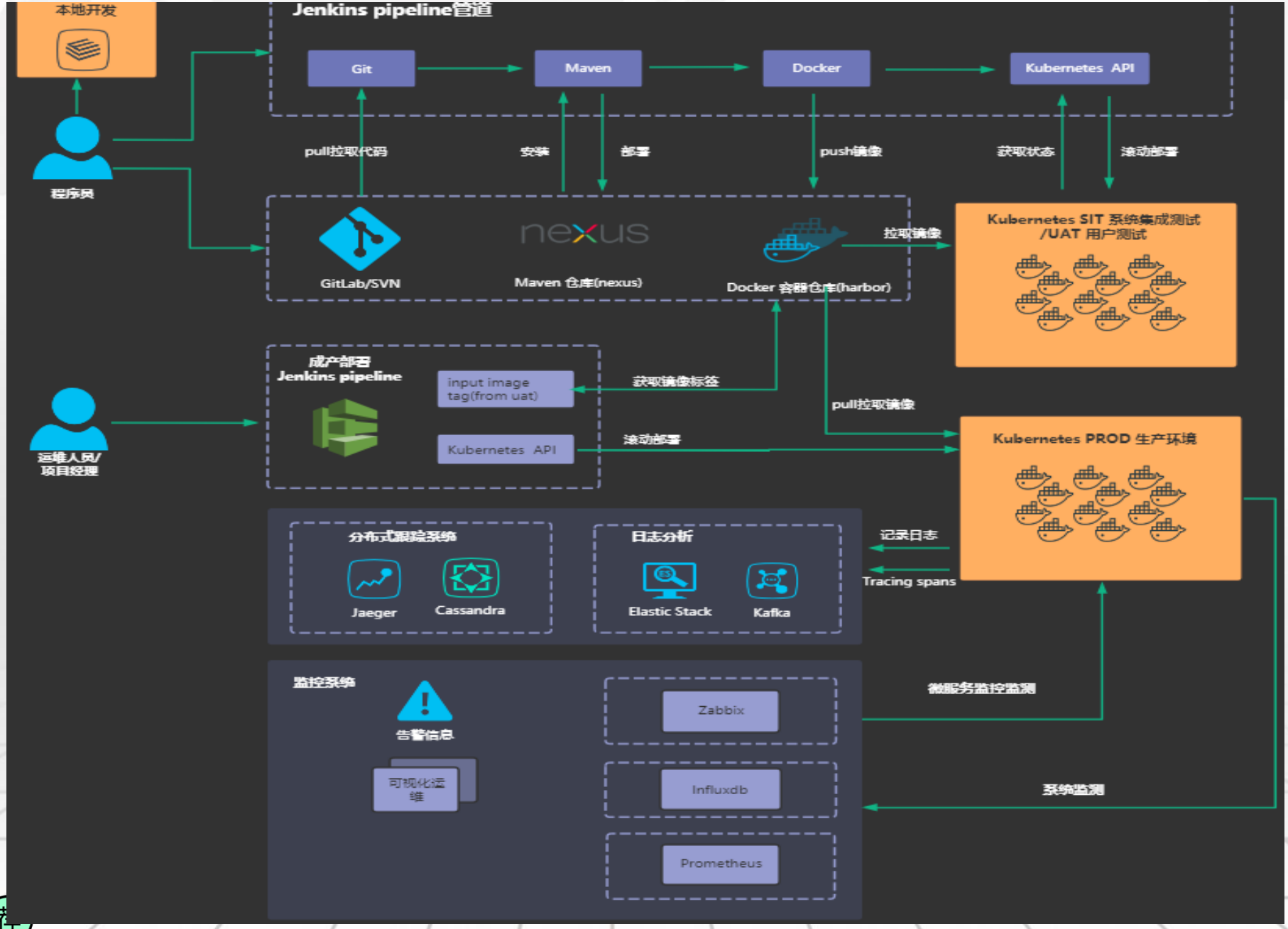
资料整理
火龙果软件
荐

# Kubectl命令

- Node、service、pod、deployment、Ingress都可以看作是资源对象，几乎所有的资源对象都可以通过kubectl来执行增、删、改、查操作，并将其保存在etcd中存储。

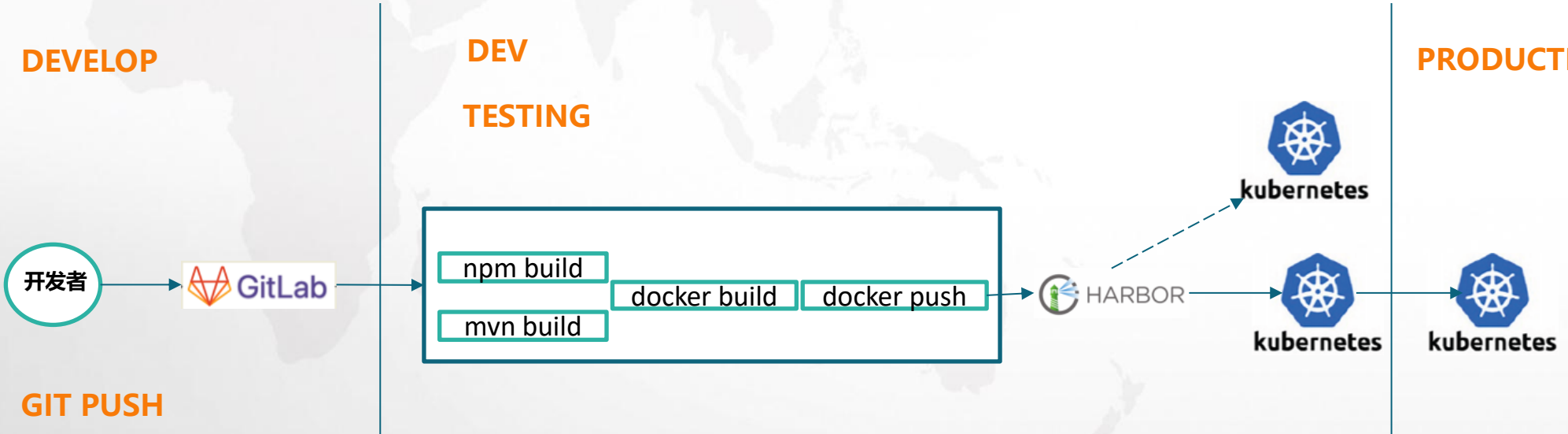
资料整理
火龙果软件
荐

# K8S容器化构建部署架构图




资料整理
火龙果软件

# K8S容器化构建部署设计




- 50+开发人员
- 每天100+次代码提交

- 每天50+次构建
- 每天新增50+docker image

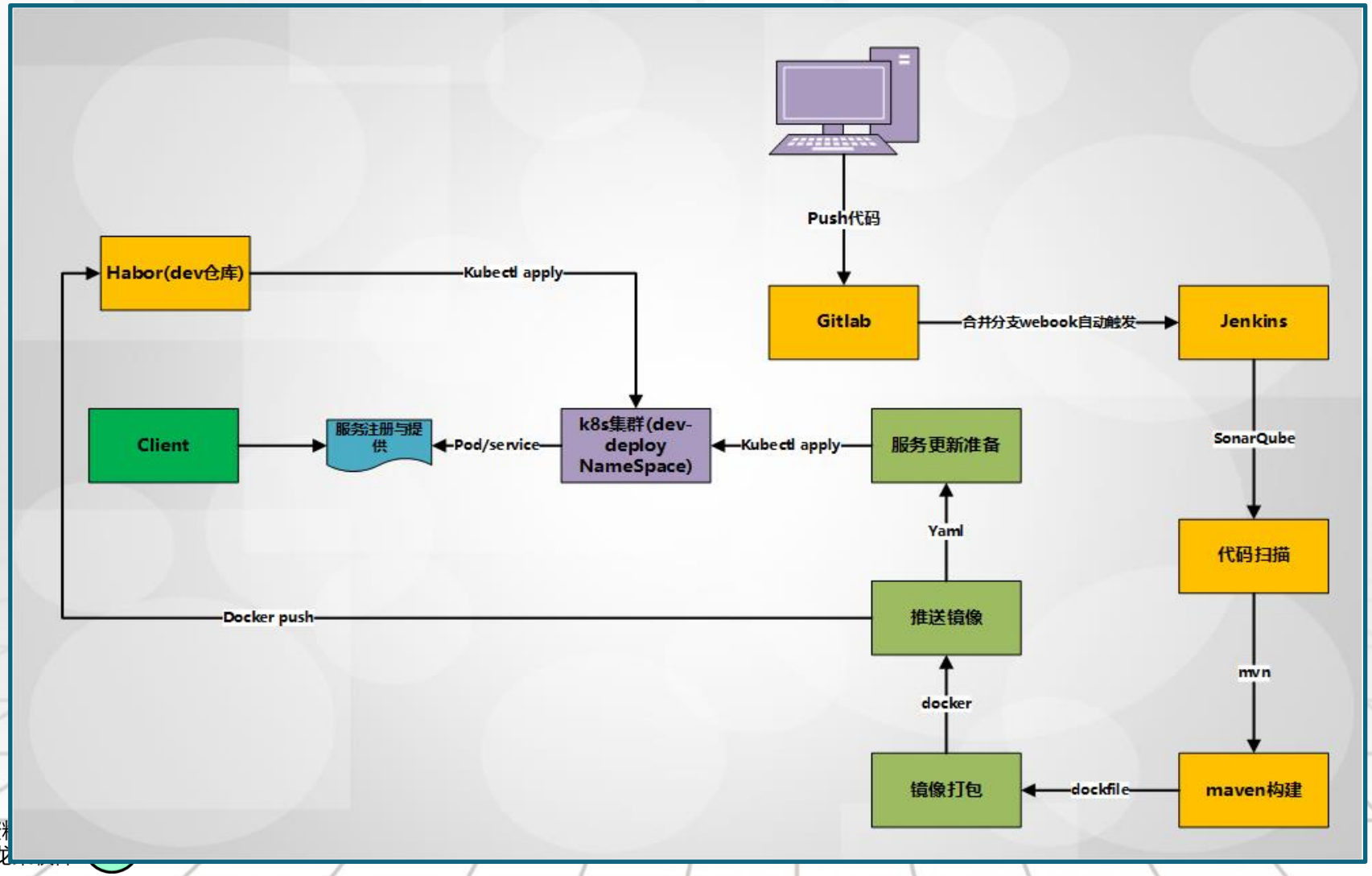
- 每天更新开发环境，测试环境至少一次
- 每个环境20+个docker容器

- 每周更新生产环境2+次

资料整理
火龙果软件
荐

# K8S容器化构建部署设计 - 开发环境

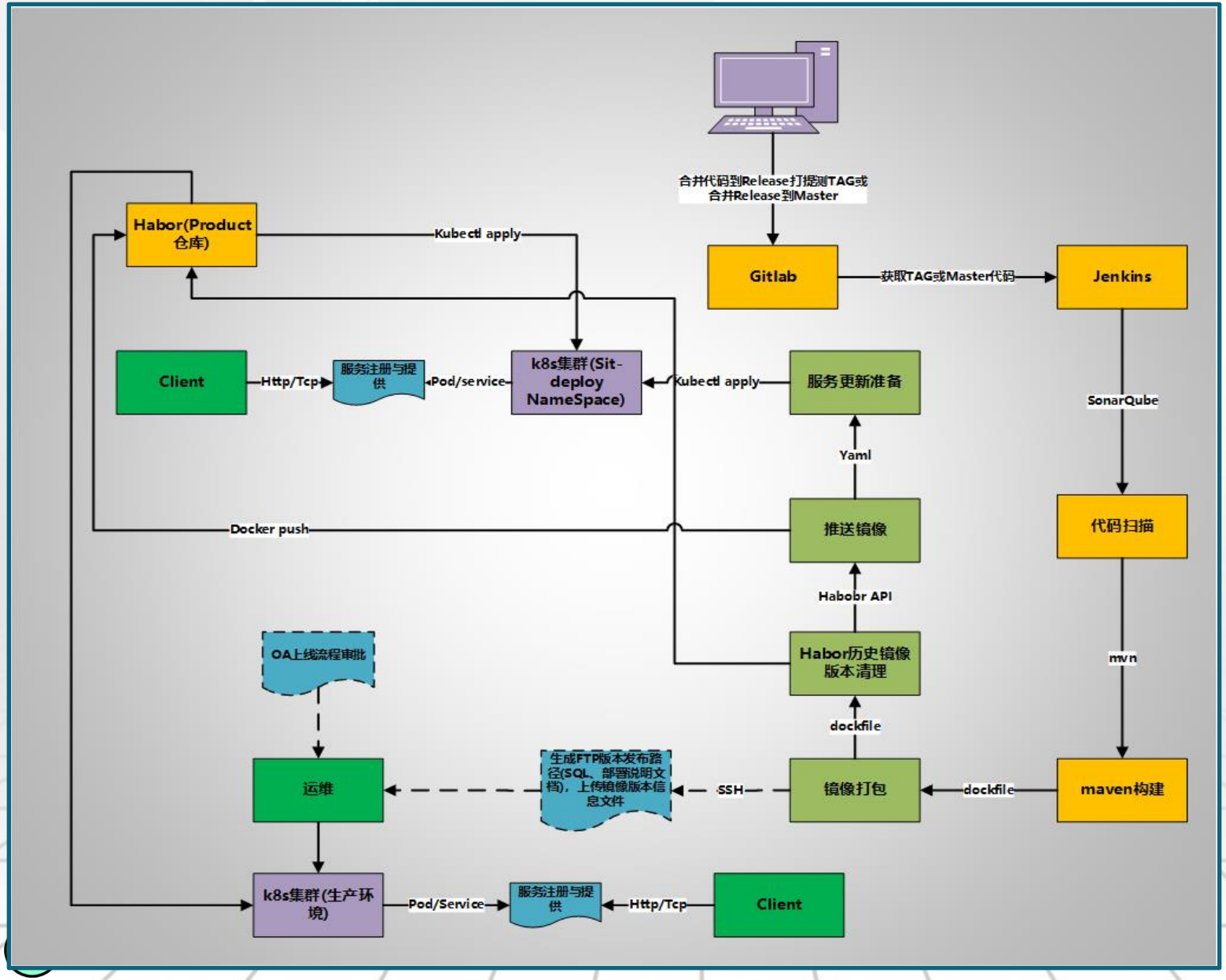
- 开发环境和测试环境使用同一套k8s集群环境(暂定)，同一个Harbor镜像仓库，使用不同的k8s命名空间来区分开发与测试环境(暂定)。
- 镜像版本TAG号由开发自行定义与升级，如1.2.0和JAR包版本号保持相同。




- 整体的交付流水线，在持续集成阶段工具链不变，依旧是Jenkins、Gitlab、Maven。
- 增加Docker和Harbor，Jenkins打通集成k8s平台，在部署与运维环节则完全由Kubernetes现实。
- 开发环境的service对外访问使用，由IP+NodePort的形式提供。
- 此开发环境默认为稳定的开发环境，所有代码统一在develop分支上。

资料
火龙

# K8S容器化构建部署设计 - 测试环境与对接运维




●测试和生产环境，使用同一个Harbor镜像仓库，由测试人员指定上线镜像版本和FTP上线路径。

●镜像TAG版本号由开发版本号+Jenkins Build Number号组成，如1.2.0.100四位组成，第四位在每次构建镜像打包后升级。

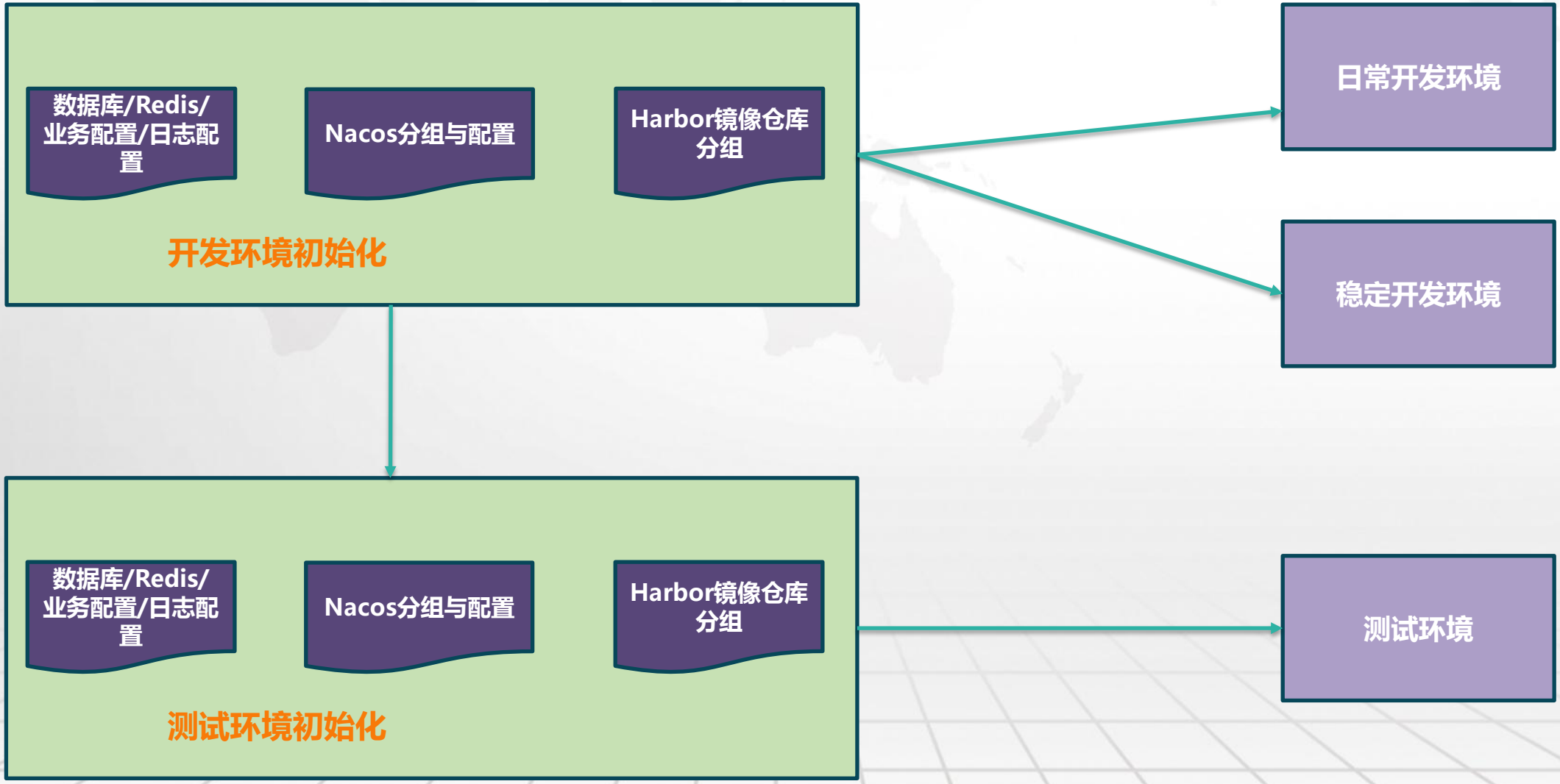
●为防止Habor仓库历史版本过多，每个仓库仅保留最近5个镜像版本。

●应用服务版本升级与回滚，K8S提供有多种方式，方式1通过拉取新的镜像版本重新生成pod，
方式2通过set image和rollout来指定版本和回滚版本。

●测试环境的service对外访问使用，由ingress组件提供的静态域名访问使用。

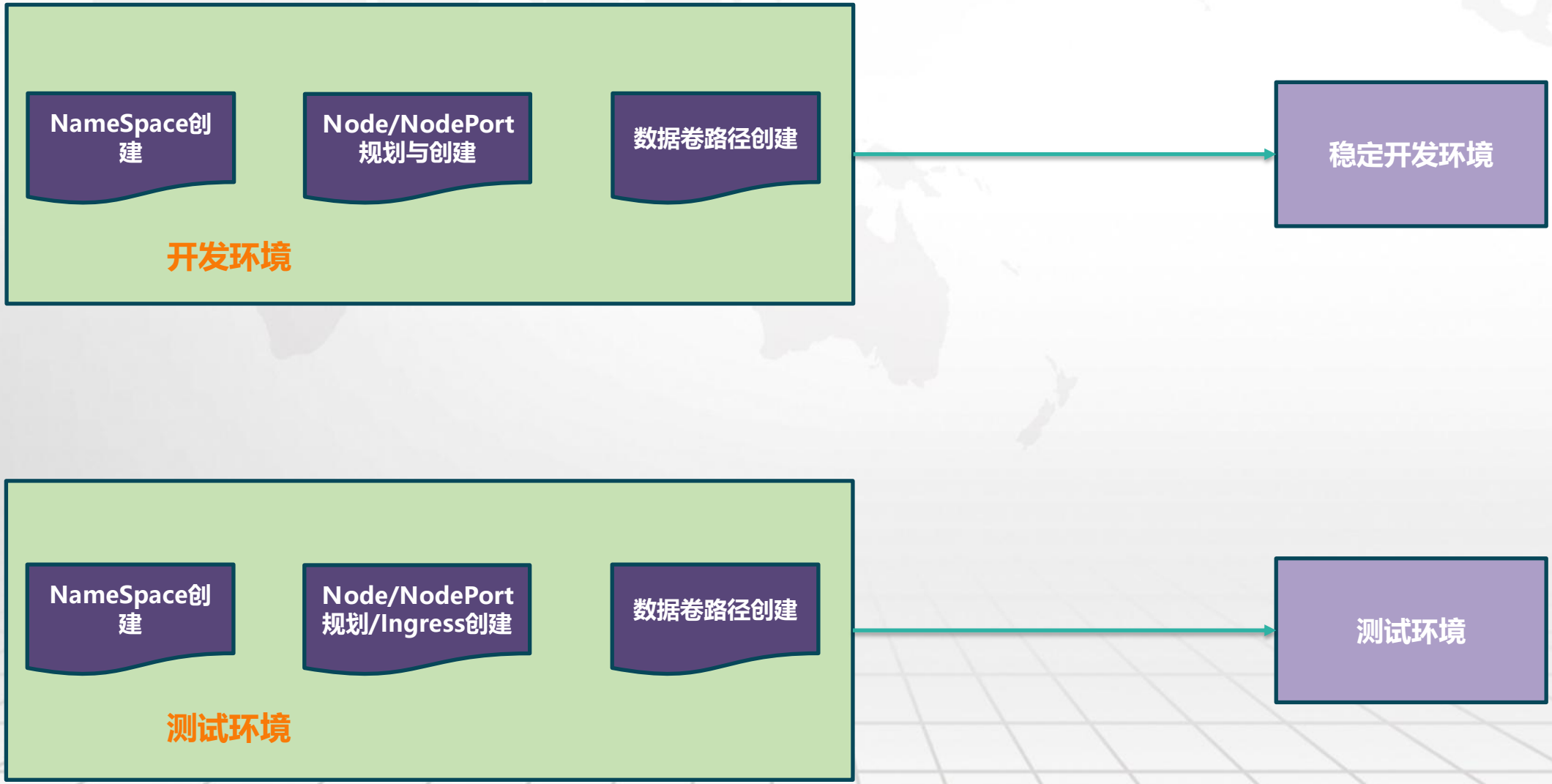
资料整理
火龙果软件

# K8S部署过程创建-环境初始化



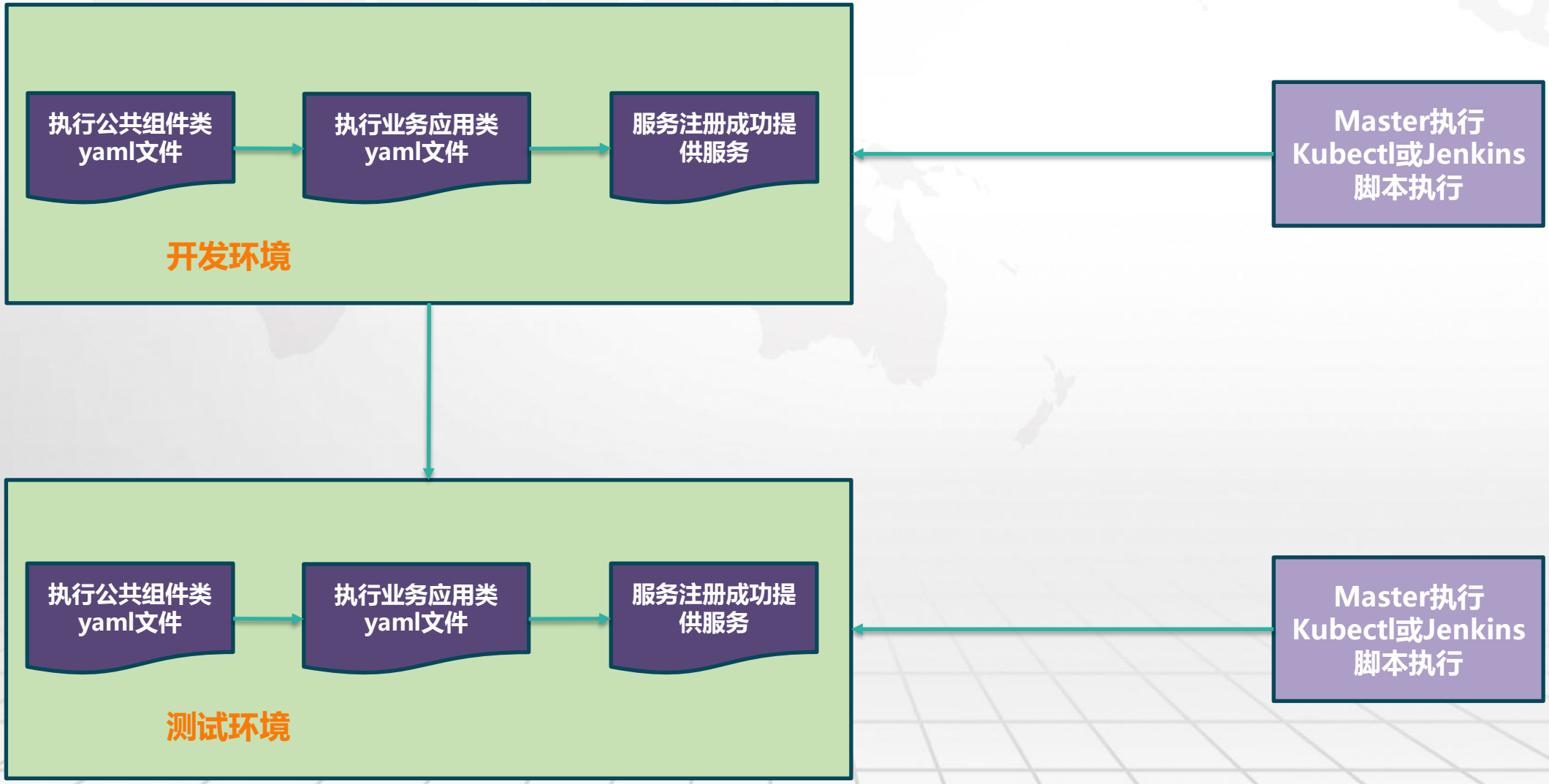

资料整理
火龙果软件
荐

# K8S部署过程创建-公共及组件类创建




资料整理
火龙果软件
荐

# K8S部署过程创建-准备与执行yaml文件




资料整理
火龙果软件
荐

# K8S部署过程创建-组件与信息查看

**Kubectl命令或Dashboard查看**

```
[root@test-kube-master ~]# kubectl get pod -n sit-deploy
NAME                                    READY   STATUS    RESTARTS   AGE
anso-uaa-645957d4fb-ks2f2               1/1     Running   0          3d15h
file-center-server-765685cb9-gln6r      1/1     Running   1          4d23h
message-center-server-6fbb4cb795-mkcc9  1/1     Running   1          4d23h
nginx-ingress-controller-ws4d8          1/1     Running   1          10d
sc-gateway-684d68d874-pxqsd             1/1     Running   0          3d20h
user-center-5895548bf5-5x6xp            1/1     Running   1          4d23h
user-center-server-5cf8dc64bd-tzfx4     1/1     Running   1          4d23h
```

资料整理
火龙果软件
荐

# K8S部署过程创建-开发环境应用服务部署过程

- 开发环境分为日常开发环境和稳定的开发环境，其中稳定的开发环境镜像由develop分支构建打包，并推送应用镜像到Harbor的dev仓库。

- 开发环境各公共类和应用服务yaml脚本准备，kubectl命令执行yaml脚本，拉取指定版本镜像生成pod，注册容器IP到开发环境Nacos注册中心，注册成功后，通过service的NodeIP提供对外访问。

- 节点机镜像版本在pod容器启动后删除，Harbor上各仓库仅保留5个最近版本。

资料整理
火龙果软件
荐

# K8S部署过程创建-测试环境应用服务部署过程

- 测试环境镜像由TAG或Master分支构建打包，并推送应用镜像到Harbor的product仓库。

- 测试环境各公共类和应用服务yaml脚本准备，kubectl命令执行yaml脚本，拉取指定版本镜像生成pod，注册容器IP到测试Nacos注册中心，注册成功后，通过service的NodeIP或Ingress提供对外访问。

- 节点机镜像版本在pod容器启动后删除，Harbor上各仓库仅保留5个最近版本。

- 测试通过后，测试人员指定Harbor的product仓库镜像版本对接给运维上线。

资料整理
火龙果软件
荐

# K8S部署过程创建-公共及组件yaml文件解析

创建NameSpace：sit-deploy.yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: sit-deploy
  labels:
    name: sit-deploy
```

创建Ingress入口组件： aliyun-ingress-nginx.yaml

直接找的aliyun的组件yaml，篇幅过长，暂不展示。

资料整理
火龙果软件
荐

# K8S部署过程创建-公共及组件yaml文件解析

创建Harbor本地仓库访问许可组件：

**在sit-deploy命名空间中，创建secret，命名为docker-reg-secret**

创建成功后，在创建pods的yaml文件中增加内容：

```
imagePullSecrets:
- name: docker-reg-secret
```

资料整理
火龙果软件
荐

# K8S部署过程创建-公共及组件yaml文件解析

创建HPA自动水平扩容组件：

```
kubectl autoscale deployment sc-gateway-deployment
--max=3 --min=1 --cpu-percent=50
```

**#为deployment资源sc-gateway-deployment创建HPA，pod数量上限为3个，最少1个，在pod平均cpu达到50%后开始自动扩容。**

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

创建Ingress应用入口： sc-gateway-ingress.yaml

**Ingress通过指定的域名访问业务应用，指定了转发的service为sc-gateway ，此服务的端口为9900，转发路径为根路径，即sc-gateway服务注册成功后，可以直接通过域名访问使用！**

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: sc-gateway-ingress
  namespace: sit-deploy
spec:
  rules:
    - host: test-kube-master
      http:
        paths:
          - backend:
              serviceName: sc-gateway
              servicePort: 9900
            path: /
```

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

**创建sc-gateway-deployment应用：sc-gateway-deployment.yaml**

Kind: Deployment，支持动态扩容，滚动升级，版本回滚。

nodeName: 指定pod运行节点在test-kube-slave02。

App: 指定应用名为sc-gateway。

Replicas: 1指定创建的pod副本数。

matchLables: 用于定义一组标签，提供给service使用。

Image: 指定镜像版本来源。

Port: 指定应用容器创建后的端口。

volumeMounts: 指定容器内挂载路径，其中/opt/config为nacos配置文件路径，由于镜像打包固定在了/opt，所以其配置文件只能配置为/opt/config。

/logs-sit/业务日志输出路径，该日志路径需要先在nacos配置中心配置好日志输出路径。

Volumes:hostPath: 数据卷使用hostpath类型，节点宿主机路径，对应映射到容器内的路径，这里应用配置路径为/opt/sit/config，日志路径为/logs-sit。

Type : Directory表示路径必需提前创建好。

imagePullSecrets: Harbor本地仓库访问许可，需要先定义与创建成功。

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

## 创建sc-gateway-deployment应用：Pod调度策略

创建pod时K8s的调度策略默认为随机状态，并选择资源最优的节点机来创建pod。

nodeName: 指定pod运行节点在test-kube-slave02，表示跳过了k8s的调度策略，直接将pod指定到了test-kube-slave02节点，但是当该节点机不存在、网络异常，或者节点资源负载压力过大时，pod将不会在该节点上创建，也不会再其它节点上创建。

nodeSelector调度：以标签的形式匹配可调度到的节点。

先给test-kube-slave01节点打上标签：Kubectl label nodes test-kube-slave01 disk-type=ssd

kubectl get nodes --show-labels | grep ssd可以查看到已经在test-kube-slave01节点上打上了标签。

sc-gateway-deployment.yaml增加调度选择：

```
nodeSelector:
    disk-type: ssd
```

执行kubectl apply -f sc-gateway-deployment.yaml -n sit-deploy后，则在已打了标签的test-kube-slave01上创建pod。

当节点没有打标签时则不会创建pod，如果有多个节点打了相同的标签，pod则会创建在多个节点上。

# K8S部署过程创建-应用yaml文件解析

**Pod容器资源限制：**

k8s采用request和limit两种限制类型来对资源进行分配request(资源需求)：即运行pod的节点必须满足运行pod的最基本需求才能运行pod，limit(资源限制)：即运行pod期间，可能内存使用量会增加，那最多能使用多少内存。

资源类型：CPU的单位是核心数，内存的单位是字节，一个pod申请0.5个CPU，就相当于申请1个CPU的一半，可以加个后缀m表示千分之一的概念。

k8s将一个逻辑cpu划分成1000个millicore(毫核)。例如 limits.cpu=500m相当于0.5个cpu，limits.cpu=2表示占用2个cpu

内存单位：K，M，G，T，P，E #通常是以1000为换算标准。

Ki,Mi,Gi,Ti,Pi,Ei #通常是以1024为换算标准。

例如：256Mi=1024*1024*256=268435456字节

256M=1000*1000*256=256000000字节

如果pod超过其内存限制，则会被终止，如果可以重新启动，则与其他所有类型的运行故障一样，kubelet将重新启动它。

如果一个pod超过其内存要求，那么当节点内存不足时，它的pod可能被逐出。

```
resources:
  limits:
    cpu: 500m
    memory: 256Mi
  requests:
    cpu: 500m
    memory: 256Mi
```

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

## nameSpace命名空间资源限制

Pod的配额，限制可以在nameSpace中运行的Pod数量。

apiVersion: v1

kind: ResourceQuota

metadata:

  name: pod-demo

spec:

  hard:

    pods: “5”

LimitRange资源限制, LimitRange 在namespace中设置的最小和最大内存限制，只有在创建和更新pod时才会被应用，改变LimitRange不会对之前创建的pod有影响。

ResourceQuota设置配额限制，namespace资源硬限制。

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
spec:
  hard:                          硬限制
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limitrange-memory
spec:
  limits:
 - default:
      cpu: 0.5
      memory: 512Mi
    defaultRequest:
      cpu: 0.1
      memory: 256Mi
    max:
      cpu: 1
      memory: 1Gi
    min:
      cpu: 0.1
      memory: 100Mi
    type: Container
```

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

## K8S中Pod的数据存储：

概念：

Volume：本质上Kubernetes Volume是一个目录，这一点与Docker Volume类似，当Volume被mount到Pod，Pod中的所有容器都可以访问这个Volume，Kubernetes Volume也支持多种类型，包括 emptyDir(生命周期和pod一致)、hostPath、GCE Persistent Disk、AWS Elastic Block Store、NFS、Ceph 等。

PV：PersistentVolume (持久化卷)，是对底层的共享存储的一种抽象，是外部存储系统中的一块存储空间，由管理员创建和维护，与Volume一样，PV具有持久性，生命周期独立于Pod，它和具体的底层的共享存储技术的实现方式有关，比如Ceph、GlusterFS、NFS等。

PVC： PersistenVolumeClaim (持久化卷声明)，PVC是用户存储的一种声明，需要为Pod分配存储资源时，可以创建一个PVC，指明存储资源的容量大小和访问模式（比如只读）等信息，Kubernetes会查找并提供满足条件的PV。PVC可以请求特定的存储空间和访问模式，对于真正存储的用户不需要关心底层的存储实现细节，只需要直接使用PVC即可。

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

## K8S中Pod的数据存储：创建PV与PVC

**使用NFS共享存储，创建nfs-pv.yaml:**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs
spec:
  storageClassName: nfs
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.207.121
    path: "/nas/dg_vd"
```

**创建nfs-pvc.yaml:**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: nfs
  resources:
    requests:
      storage: 1Gi
```

**Pod中使用，创建nginx-deployment.yaml:**

```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: testpv
  labels:
    role: web-frontend
spec:
  containers:
  - name: web
    image: nginx
    ports:
      - name: web
        containerPort: 80
    volumeMounts:
        - name: nfs
          mountPath: "/usr/share/nginx/html"
  volumes:
  - name: nfs
    persistentVolumeClaim:
      claimName: nfs
```

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

创建sc-gateway-service：sc-gateway-service.yaml

Kind: service，资源对象为service服务。

labels:app: 指定service标签名为sc-gateway。

Metadata:name: 指定service名为sc-gateway。

namespace: 指定命名空间为sit-deploy。

Port:Name: service端口名称。

Port: service的Cluster IP上的端口，即service自己的端口。

targetPort: service对应的podIP上端口，即pod端口。

nodePort : service服务暴露给节点机对外访问的端口。

slector: 指定为sc-gateway标签的应用进行负载均衡。

Type: service服务的暴露类型。

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: sc-gateway
  name: sc-gateway
  namespace: sit-deploy
spec:
  ports:
    - name: sc-gateway
      port: 9900
      targetPort: 9900
      nodePort: 19900
  selector:
    app: sc-gateway
  type: NodePort
```

资料整理
火龙果软件
荐

# K8S部署过程创建-应用yaml文件解析

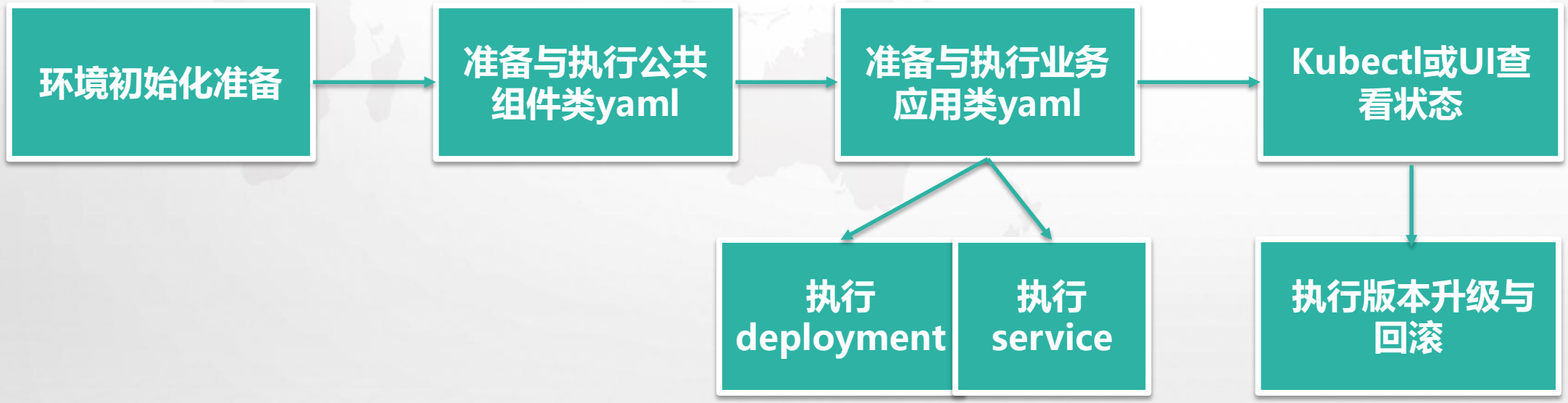
- **sc-gateway服务为路由转发服务，**提供所有服务的统一入口，在k8s上，只需将该服务的端口暴露出去提供访问即可使用。

也就是说权限框架的其它服务，在容器启动后，容器内部网络畅通，各服务无需再创建service资源，只需要创建sc-gateway的service和对外访问即可。

- **通过创建sit-deploy和dev-deploy命名空间，**以及挂载不同的配置文件和日志文件路径来区分不同的部署环境。

资料整理
火龙果软件
荐

# K8S部署过程创建-总体流程




资料整理
火龙果软件
荐

# K8S部署过程创建-应用查看与使用

上面，我们举例创建了一个权限框架中的sc-gateway服务，我们可以登录到10.0.0.69:30443 去查看这个应用的相关信息，比如pod、deployment、service等等。

然后我们可以以此为例创建更多的应用服务，比如权限框架的所有应用服务等。

当sc-gateway服务或anso-uaa等服务正常创建并已提供对外访问后，我们就可以和以往一样直接使用该服务提供的接口功能了。

sit-deploy　搜索

lancing > Services

| 名称 | 标签 | 类别 | 集群 IP | 内部 Endpoints | 外部 |
|---|---|---|---|---|---|
| anso-uaa | app: anso-uaa | NodePort | 10.106.56.41 | anso-uaa.sit-deploy:8000 TCP<br>anso-uaa.sit-deploy:18000 TCP | - |
| message-center-server | app: message-center-server | NodePort | 10.105.199.233 | message-center-server.sit-deploy:6100 TCP<br>message-center-server.sit-deploy:16100 TCP | - |
| nginx-ingress-lb | app: ingress-nginx | ClusterIP | 10.106.52.113 | nginx-ingress-lb.sit-deploy:80 TCP<br>nginx-ingress-lb.sit-deploy:0 TCP<br>nginx-ingress-lb.sit-deploy:443 TCP<br>nginx-ingress-lb.sit-deploy:0 TCP<br>nginx-ingress-lb.sit-deploy:10254 TCP<br>nginx-ingress-lb.sit-deploy:0 TCP | - |
| file-center-server | app: file-center-server | NodePort | 10.107.233.253 | file-center-server.sit-deploy:5000 TCP<br>file-center-server.sit-deploy:15000 TCP | - |
| user-center-server | app: user-center-server | NodePort | 10.98.66.22 | user-center-server.sit-deploy:7000 TCP<br>user-center-server.sit-deploy:17000 TCP | - |
| user-center | app: user-center | NodePort | 10.104.119.122 | user-center.sit-deploy:6000 TCP<br>user-center.sit-deploy:16000 TCP | - |
| sc-gateway | app: sc-gateway | NodePort | 10.107.242.92 | sc-gateway.sit-deploy:9900 TCP<br>sc-gateway.sit-deploy:19900 TCP | - |

资料整理
火龙果软件
荐

# K8S部署过程创建-应用服务版本升级与回滚

●在应用yaml文件解析中，有说明到Image:指定镜像版本来源，对于开发环境镜像版本会一直覆盖使用，如1.2.0，对于测试环境则增加一位Jenkins Build Number号，而镜像仓库仅保留5个最近版本，回滚操作可以由jenkins提供的回滚功能实现。

●实际上，测试环境发包依旧可以通过选择不同的TAG来打包为可用的镜像版本。

● Kubectl命令执行版本升级和回滚，一种是修改yaml中的image版本后重新执行yaml文件。

●另一种是使用k8s自身的回滚命令实现。

资料整理
火龙果软件
荐

# K8S部署过程创建-常见kubectl命令

#查看集群信息：kubectl get cs

#查看node节点信息：kubectl get node

#打指定节点的标签：kubectl label node test-kube-slave01 disk-type=ssd

#查看labels标签列表：kubectl get nodes --show-labels | grep ssd

#查看pod列表： kubectl get pods -n sit-deploy -o wide

#查看指定pod详细信息：kubectl describe pod sc-gateway-684d68d874-pxqsd -n sit-deploy

#创建deployment资源对象：kubectl create –f sc-gateway-deployment.yaml –n sit-deploy

kubectl apply –f sc-gateway-deployment.yaml –n sit-deploy (更新创建)

#重启指定pod(重启并增加副本数)： kubectl scale deployment sc-gateway-deployment --replicas=2

#强制更新pod(先删除后创建)：kubectl replace –force –f sc-gateway-deployment.yaml –n sit-deploy

#查看service列表：kubectl get svc -n sit-deploy

#查看service详细信息：kubectl describe svc sc-gateway -n sit-deploy

#查看deployment列表：kubectl get deployment -n sit-deploy

#查看deployment详细信息：kubectl describe deployment sc-gateway -n sit-deploy

#查看指定pod日志信息： kubectl logs -f sc-gateway-cc4c78597-f9lng -n sit-deploy --tail=200

#删除指定pod，先删除对应的deployment再删除pod： kubectl delete deployment uaa-server -n sit-deploy

kubectl delete -n sit-deploy --force pod uaa-server-78f4d5d6d4-s7cft

资源对象已存在时，导出yaml文件：kubectl get svc production -o yaml -n gs-deploy >production_bak.yaml

以create命令导出后可以修改yaml后再创建资源对象 ：kubectl create ns ns-test -o yaml --dry-run=client >ns-test.yaml

资料整
火龙果

# K8S部署过程创建-灰度发布

概念：

灰度发布：又名金丝雀发布，简单的讲是一套A/B TEST系统；是指在新旧版本之间，能够平滑过度的一种发布方式，即让一部分流量继续使用老版本，一部分流量开始使用新版本，如果验证新版本流量没有问题，则逐步扩大范围，直至把所有流量都迁移到新版本上来，灰度发布可以保证整体系统的稳定，在初始灰度时可以发现和解决问题，并保证其影响程度。

采用ingress-Nginx组件来实现灰度发布过程，因暂无实际应用服务，找的一个Nginx应用来演示。

实现，基于权重的灰度发布。

实现，基于http header的灰度发布。

资料整理
火龙果软件
荐

# K8S部署过程创建-灰度发布实现过程

资料整理
火龙果软件
荐