中间件容器化通用步骤

一、部署应用至K8S的通用步骤【如何部署一个应用至K8S】

1.1、必须了解你部署的这个东西

- 架构
- 配置
- 端口
- 启动命令

1.2、要有这个东西的镜像

- 公司服务镜像自己做
- 中间件，例如redis可以使用官网的，自己做的未必有别人的好

1.3、找到最合适的部署方式

- 确定是否是有状态的服务【一般jar的java的java程序是无状态的】
    - 确定部署的编排方式
- 配置文件分离
    - cm、环境变量都可以，主要是看程序是否支持
- 部署文件来源
    - yaml文件是公司的自己写
    - 中间件可以参考官网的
- 如何部署
    - 集群------> sts、Operator
    - 单机-------> deploy

1.4、这个程序如何被使用

- 使用的协议
    - http
- 内部还是外部
    - nodeport

二、中间件单实例部署

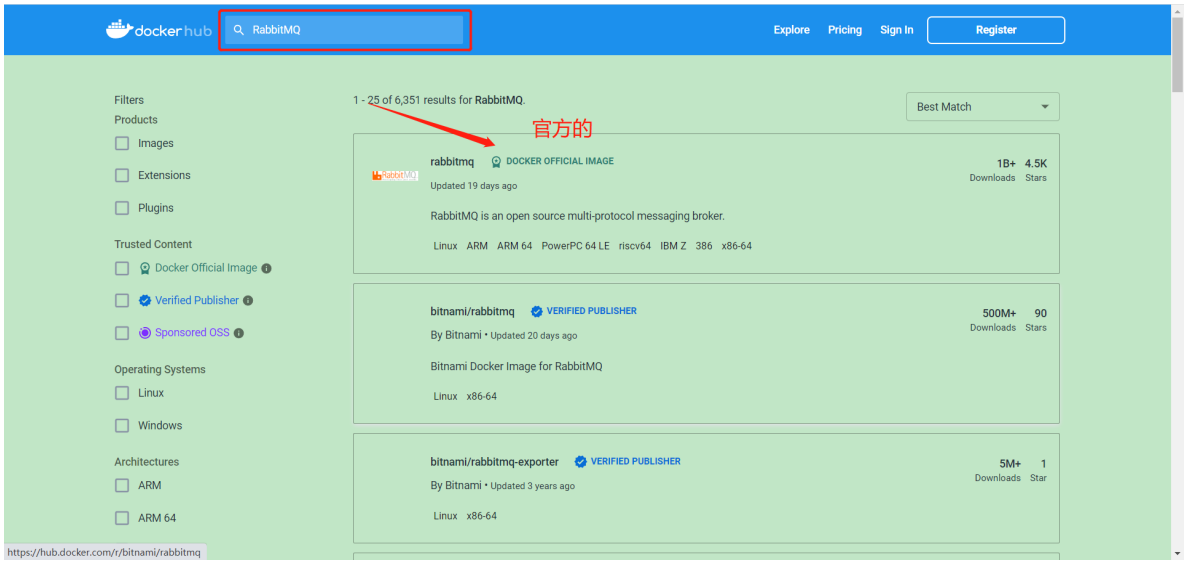
2.1、找到官方镜像

- 在官网可以看到该中间件是否支持环境变量方式读取配置文件、支持哪个环境变量、配置文件路径、启动命令、端口、架构等

```
# 官网
https://hub.docker.com/

# rabbitmq为例部署
https://hub.docker.com/_/rabbitmq

# 镜像地址选择
https://hub.docker.com/_/rabbitmq/tags
rabbitmq:3.8.33-management
```



2.2、确认需要部署的配置：环境变量或者配置文件

- 在官网可以看到该中间件是否支持环境变量方式读取配置文件、支持哪个环境变量、配置文件路径、启动命令、端口、架构等
- 有环境变量支持的说明可以通过注入环境变量的方式配置该应用的配置文件

2.3、选择部署方式

- 单机测试可以使用deploy的方式去部署

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rabbitmq
```

```yaml
  namespace: infra
  labels:
    app: rabbitmq
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq
  strategy:           # 滚动更新配置
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: rabbitmq
    spec:
      containers:
      - image: rabbitmq:3.8.33-management
        name: rebbitmd
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 5672
          name: web
          protocol: TCP
        env:                # 环境变量, 其他支持的环境变量查看hub.docker官网镜像简介
        - name: TZ
          value: Asia/Shanghai
        - name: LANG
          value: C.UTF-8
        - name: RABBITMQ_DEFAULT_USER
          value: root
        - name: RABBITMQ_DEFAULT_PASS
          value: root123
        lifecycle: {}       #  生命周期配置
        livenessProbe:      #  健康检查
          failureThreshold: 2
          initialDelaySeconds: 10
          successThreshold: 1
          tcpSocket:
            port: 5672
          timeoutSeconds: 2
        readinessProbe:
          failureThreshold: 2
          initialDelaySeconds: 10
          successThreshold: 1
          tcpSocket:
            port: 5672
          timeoutSeconds: 2
        resources:
          limits:
            cpu: 998m
            memory: 1019Mi
          requests:
            cpu: 998m
            memory: 1019Mi
      affinity: {}              # 亲和性配置
```

```
      dnsPolicy: ClusterFirst    # 采用集群DNS 默认就是这个
      restartPolicy: Always      # Pod重启策略
```

2.4、访问配置TCP或者HTTP

- 创建SVC、ingress都可以

```
apiVersion: v1
kind: Service
metadata:
  name: rabbitmq
  namespace: infra
spec:
  ports:
  - name: web
    port: 5672
    protocol: TCP
    targetPort: 5672
  - name: http
    port: 15672
    protocol: TCP
    targetPort: 15672
  selector:
    app: rabbitmq
  sessionAffinity: ClientIP
  type: NodePort
```

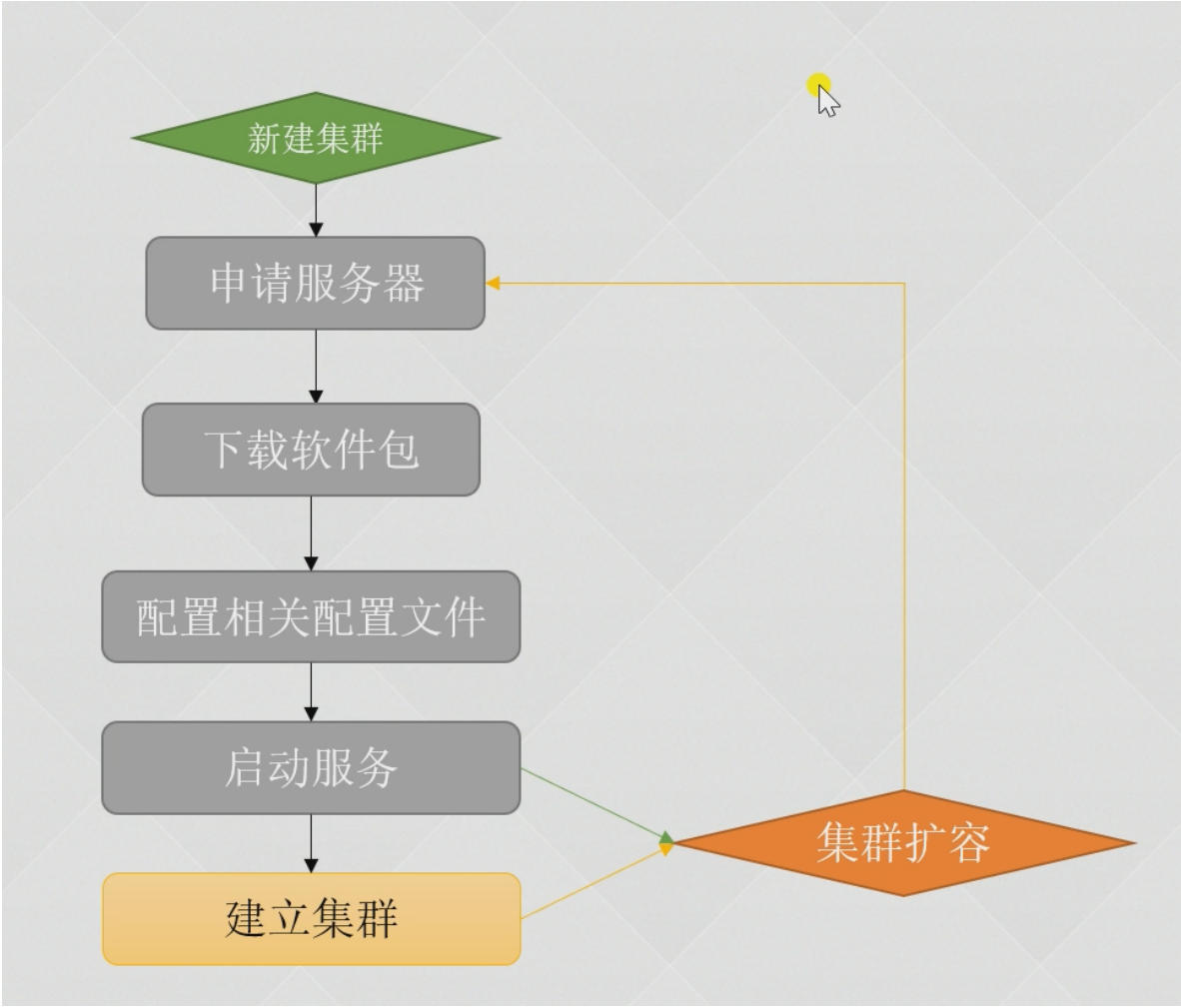
- 查看部署结果【再次apply滚动更新】

```
[root@k8s-master01 单机]# kubectl get po  -n infra -l app=rabbitmq
NAME                        READY   STATUS              RESTARTS   AGE
rabbitmq-5c99f4bfb7-6mg4x   1/1     Running             0          6m53s
rabbitmq-69d79498d5-vg5mz   0/1     ContainerCreating   0          30s

[root@k8s-master01 单机]# kubectl get svc -n infra rabbitmq
NAME       TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)
     AGE
rabbitmq   NodePort   10.107.175.251   <none>
 5672:31884/TCP,15672:31531/TCP   62m
```

三、k8s跟传统应用管理中间件的区别

3.1、传统应用管理中间件



3.2、k8s管理中间件集群

包管理工具：一句话总结功能就是可以很方便管理一些比较复杂的应用，比如MySQL集群、Redis集群等，可以一键式创建集群、扩容、备份等。常用的两种包管理工具是Operator和Helm。

- Helm：更倾向于无状态应用的部署，比如公司的服务、某些不需要持久化数据的中间件、不需要实现额外功能的服务，比如备份、回滚等功能
- Operator：管理更为复杂的有状态服务，比如MySQL集群、Redis集群、Rook等。并且可以利用Operator实现扩容、备份、回滚等功能

四、中间件到底要不要部署到k8s

- 非生产环境：使用K8s管理比较推荐
- 生产环境：需要考虑性能、持久化、稳定性等问题
- all in to k8s