

k8s & docker

[使用minikube部署mongodb和mongosh](#)

[k8s Operator](#)

用kubernetes去管理Docker集群，即可以将Docker看成Kubernetes内部使用的低级别组件。

<https://www.jianshu.com/p/a7d25b6485da>

 [dockerfile 与 docker-compose的区别](#)

 [最详细记录minikube部署第一个Kubernet应用_查理曼大帝的博客-CSDN博客](#)

 [Translate a Docker Compose File to Kubernetes Resources](#)

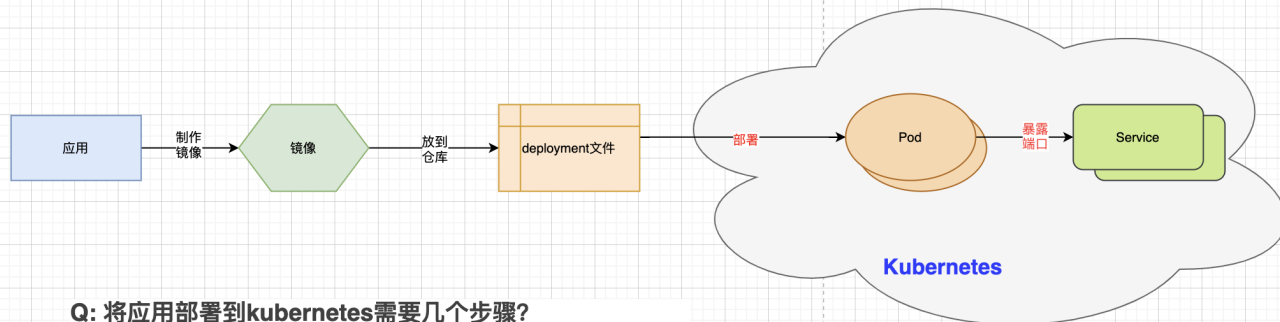
镜像构建：即创建一个镜像，它包含安装运行所需的环境、程序代码等。这个创建过程就是使用 dockerfile 来完成的。

容器启动：容器最终运行起来是通过拉取构建好的镜像，通过一系列运行指令（如端口映射、外部数据挂载、环境变量等）来启动服务的。针对单个容器，这可以通过 docker run 来运行。

而如果涉及多个容器的运行（如服务编排）就可以通过 docker-compose 来实现，它可以轻松的将多个容器作为 service 来运行（当然也可仅运行其中的某个），并且提供了 scale (服务扩容) 的功能。

Dockerfile 记录单个镜像的构建过程， docker-compse.yml 记录一个项目(project, 一般是多个镜像)的构建过程。

Q: 将应用部署到kubernetes需要几个步骤?



Q: 将应用部署到kubernetes需要几个步骤?



A: 在kubernetes中部署应用，首先需要将应用做成镜像，然后才能部署在kubernetes中。

所以：

- 创建一个镜像
- 需要一个镜像仓库用来存放镜像
- 编写kubernetes的deployment文件
- 将镜像部署成为pod
- 编写kubernetes的service文件，创建pod的服务，并对外暴露端口

使用minikube部署mongodb和mongosh

<https://www.jianshu.com/p/8beabf87daf0>

1. 参考  minikube start 安装和启动minikube
2. mongodb的环境变量需要定义MongoDB的username和password
 - a. 使用 `echo -n 'xxxxx' | base64` 输出自定义的用户名和密码加密过后的字符串，写入secret组件（通常用来存放敏感信息  Secret）
3. 定义secret.yaml

▼ mongodb-secret.yaml

YAML |  复制代码

```
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mongodb-secret
5  type: Opaque
6  data:
7    mongo-root-username: b3BzQWRtaW4= # admin
8    mongo-root-password: MTIzNDU2 # 123456
```

- apiVersion和kind这两行声明是必须的，因为要定义Secret组件，那么kind就是Secret。

- metadata中的name是Secret运行后的名字，可以自由定义。
 - type是Secret的type，Opaque是最常见的type，base64编码格式的Secret，用来存储密码、秘钥等。
 - data就是Secret的内容。data需要用base64加密后，再cp到这里。
4. 使用 `kubectl apply -f ./mongodb-secret.yaml` 创建secret组件
 5. 使用 `kubectl get secrets` 命令查看是否创建成功
 6. 编写deployment.yaml和services.yaml，这里写在了一起

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mongodb-deployment
5    labels:
6      app: mongodb
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: mongodb
12    template:
13     metadata:
14       labels:
15         app: mongodb
16     spec:
17       containers:
18       - name: mongodb
19         image: mongo
20         ports:
21         - containerPort: 27017
22       env:
23       - name: MONGO_INITDB_ROOT_USERNAME
24         valueFrom:
25           secretKeyRef:
26             name: mongodb-secret
27             key: mongo-root-username
28       - name: MONGO_INITDB_ROOT_PASSWORD
29         valueFrom:
30           secretKeyRef:
31             name: mongodb-secret
32             key: mongo-root-password
33 ---
34 apiVersion: v1
35 kind: Service
36 metadata:
37   name: mongodb-service
38 spec:
39   selector:
40     app: mongodb
41   ports:
42   - protocol: TCP
43     port: 27017
44     targetPort: 27017
```

关于service的说明（下半部分）：

- apiVersion和kind：最前面的两行声明，kind表示定义的组件是Service。
 - metadata和name表示Service的名称，可自行取名。
 - selector：和上述Deployment定义中的Pod联系起来。（selector中的和上述template中的metadata中的labels对应）。
 - ports：port表示Service的端口，targetPort表示Pod中Container容器的端口，与上述的Deployment中的Pod中的Container的containerPort要一致。port和targetPort可以不一致。
7. 使用 `kubectl apply -f ./mongodb-deployment.yaml` 创建deployment和service
 8. 使用 `kubectl get services` 和 `kubectl get deployments` 查看是否创建成功
 9. 查看Pod和Service具体信息，可以看到Service的Endpoint信息，指向的IP是Pod的IP
 10. 使用 `kubectl exec -it <podName> -- bash` 进入创建的mongo，运行/bin/mongosh命令即可连上mongodb
 - a. 使用在secret组件中定义的账号密码登录可以进行一系列的操作 `/bin/mongosh mongodb://127.0.0.1:27017 -u admin -p 123456`
 - b. 不使用在secret组件中定义的账号密码登录进行操作时会提示没有权限 `/bin/mongosh mongodb://127.0.0.1:27017`

k8s Operator

 Operator 模式

 Kubernetes Operator 快速入门教程

 Kubernetes Operator简介 – DockOne.io

<https://www.mongodb.com/docs/kubernetes-operator/master/kind-quick-start/>

Kubernetes 1.7 版本以来就引入了自定义控制器的概念，该功能可以让开发人员扩展添加新功能，更新现有的功能，并且可以自动执行一些管理任务，这些自定义的控制器就像 Kubernetes 原生的组件一样，Operator 直接使用 Kubernetes API进行开发，也就是说他们可以根据这些控制器内部编写的自定义规则来监控集群、更改 Pods/Services、对正在运行的应用进行扩缩容。

Operator可以看作是一种解决数据库等复杂应用容器化问题的模式。Operator遵循Kubernetes声明式API和Controller的设计理念，被用来扩展Kubernetes API，利用定制资源管理应用及其组件，对复杂的有状态应用，如数据库、缓存和监控系统等，进行创建、配置和管理。Operator 基于Kubernetes的资源 and 控制器概念之上构建，但同时又包含了应用程序特定的领域知识。

Operator作为解决复杂应用容器化的一种思路，通过将专业领域知识注入Kubernetes完成对复杂有状态应用的自动部署和管理。例如，在对于数据类有状态应用，不同的数据库如Redis、MySQL等，其部署、扩缩容、备份方法各有区别，Operator针对这些复杂的场景进行专业的处理。简单理解，Operator=CRD+Controller，CRD完成资源的定义，Controller监听CRD的CRUD事件根据业务逻辑进行自定义处理。

创建自定义Operator需要如下资源：

- Custom Resource (CR) spec，定义我们要观测的应用对象，以及为CR定义的API
- Custom Controller，用来观测CR
- Custom Code，决定Custom Controller如何协调CR
- Operator 框架，管理Custom Controller
- Deployment，定义Operator和自定义资源

上述内容核心的两步是CR spec的设计和controller的开发。CR spec反映应用具体的需求，如对于redis集群来说，redis集群的分片数、每个分片的从副本数、每个Redis节点的内存大小等均是redis自定义资源的spec部分。controller解析CR spec，协调调用Kubernetes内建资源（Statefulset、ConfigMap、Service等）和特定应用处理逻辑，实现对容器化应用的自动化管理。