红帽OpenShift 4 资源和工具（8小时）

工程和应用

本模块涵盖了OpenShift项目并演示了应用部署过程中的主要步骤。

你将学习如何检查应用程序的健康状况以及如何扩展应用程序。

本模块还包括YAML Manifest、其结构和主要组件。最后，你将学习如何创建和编辑YAML Manifest。

1. OpenShift® 工程

什么是工程？

- 分组你的应用程序资源的机制
  - 为资源创建命名空间
    - 资源名称在项目中必须是唯一的
  - 在每个项目的基础上创建资源配额
  - 管理项目内和项目间的访问策略
- 在Kubernetes中也被称为 "命名空间"（略有不同）。

操作工程

- 创建工程：
  - `oc new-project NAME --display-name=DISPLAYNAME --description=DESCRIPTION`
  - 自动生成此默认项目
- 列出现有项目：
  - `oc projects`（加 `-q` 参数只列出工程名）
- 设定默认工程：
  - `oc project NAME`
- 显式地指定当前工程：
  - `oc -n NAME <rest of command>`

2. 应用部署步骤

从现有镜像部署应用

- 找到要使用的容器镜像
  - 包括镜像仓库，镜像名称，标签
- 配置参数
  - ConfigMaps 和 secrets
- 持久化卷（Persistent Volumes）
- 资源请求和限制

从代码部署应用

- 基础镜像
- 应用源代码

- 构建指令
- 容器镜像仓库
- 遵循现有镜像部署应用流程

应用的可用性和可扩展性

- ReplicaSet 或 ReplicationController
  - 负责运行预定义数量的pod
- 在与配置的数量不匹配的情况下，新的pod会启动
- 在应用运行时可以改变pod的数量
  - ReplicaSet停止或启动pod以匹配配置的数量
  - 降低到零可以让应用暂停

Accessing Application

访问应用

- Pods可以启动、停止、重启，每次获得新的IP地址
- 服务为Pod组提供稳定的IP地址和DNS名称
  - 使用标签和选择器定义pod组
- 默认服务名称：`_service-name.project_.svc.cluster.local`
- 使用round-robin算法在服务后进行Pod的负载平衡
- 检查服务：`oc get svc`
- 只有集群内的pod可以访问服务

`oc new-app` **命令**

- 执行应用部署小节描述的所有步骤
- 通过许多不同的方式来部署应用程序：
  - 从容器镜像
  - 从源代码（OpenShift使用合适的镜像）
  - 从基础镜像 + 源代码
  - 从OpenShift模板
  - 从基础镜像 + 二进制包
- 运行`oc new-app -h`以找到所有可用的选项

对应用程序的外部访问

- 使用路由对应用程序进行外部访问
- 使用 `oc expose` 命令来创建路由
- 使用`oc create route` 命令创建 安全 路由（HTTPS）。
- 路由使用服务来寻找 *pod*的信息
  - 不转发流量 到服务
- 路由使用 HAProxy 来提供更复杂的负载平衡算法

3. 应用健康

健康检查

- 就绪探针

检查它所在的容器是否准备好为请求提供服务。

- 如果容器没有通过就绪测试，它将从端点列表中删除。

例如，可以控制哪些Pod被使用。

- 存活探针

检查它被安排在其中的容器是否仍在运行。

- 如果容器没有通过存活性测试，kubelet就会杀死它。
    - 下一步取决于其重启策略。

健康检查的类型

- **HTTP检查**：如果HTTP请求返回的代码在200和399之间，则为成功。
    - 适合于网络应用
- **容器执行检查**：在容器中执行命令，退出代码为0表示成功
    - 适合基本的有效性测试
- **TCP 套接字检查**：Kubelet尝试打开通往容器的TCP套接字
    - 适用于非HTTP应用

4. 应用扩缩容

- 对于横向扩展，在ReplicaSet中添加/删除类似的pods
    - 使用`oc scale --replicas=`命令
- 使用`--replicas=0`来暂停应用
- 缩放可以是自动的
    - 使用`oc autoscale`命令

5. YAML Manifest

- 有助于创建可重复的应用配置
- 包含对OpenShift资源的描述
- 可由源代码管理（SCM）系统管理

6. YAML Manifest 组件

- apiVersion:
- kind:
- metadata:
- spec:

apiVersion

- 使用`oc api-versions`命令来列出所有版本。
- 例子（截取）：

```
. . . .
apps.openshift.io/v1
apps/v1
apps/v1beta1
apps/v1beta2
. . . .
```

kind

- 使用`oc api-resources`命令来列出所有资源类型
- 例子（截取）：

```
. . . .
NAME                                SHORTNAMES       APIGROUP
                 NAMESPACED   KIND
bindings
                true         Binding
componentstatuses                   cs
                false        ComponentStatus
configmaps                          cm
                true         ConfigMap
endpoints                           ep
                true         Endpoints
events                              ev
                true         Event
limitranges                         limits
                true         LimitRange
namespaces                          ns
                false        Namespace
nodes                               no
                false        Node
persistentvolumeclaims              pvc
                 true         PersistentVolumeClaim
persistentvolumes                   pv
                false        PersistentVolume
pods                                po
                true         Pod
. . . .
```

metadata

- `name`：资源的名称
- `namespace`：资源所在的名称空间
- `labels`：识别元数据，由选择器和过滤器使用。
- `annotations`：非识别性元数据，由库和客户使用
- `ownerReferences`：拥有该资源的资源
  - 例如。ReplicationController归DeploymentConfig所有
  - 由OpenShift自动关联

spec

- 使用`oc explain`来查找有关OpenShift对象的信息

- 例如：
  - `oc explain pod`
  - `oc explain pod.spec`
  - `oc explain pod.spec.containers`
  - `oc explain pod.spec.containers.readinessProbe`
  - `oc explain pod.spec.contains.readinessProbe.httpGet`
- 或者使用`oc explain pod --recursive`来列出整个结构

7. YAML 基础

Mapping

- 简单映射

```
  name: postgresql
```

- 嵌套映射

```
  labels:
    app: django-psql-example
    template: django-psql-example
```

Lists

- 容器列表实例：

```
    containers:
    - name: logging-sidecar
      image: docker.io/busybox:latest
      volumeMounts:
      - mountPath: /tmp
        name: tmp
    - name: logtofile
      image: docker.io/wkulhanek/logtofile
      volumeMounts:
      - mountPath: /tmp
        name: tmp
    [...]
```

- 模板参数列表示例：

```
parameters:
- description: The name assigned to all of the frontend objects defined in
this template.
  displayName: Name
  name: NAME
  required: true
  value: django-psql-example
- description: The OpenShift Namespace where the ImageStream resides.
  displayName: Namespace
  name: NAMESPACE
  required: true
  value: openshift
```

```
- description: Version of Python image to be used (3.6 or latest).
  displayName: Version of Python Image
  name: PYTHON_VERSION
  required: true
  value: "3.6"
```

- OpenShift对象的复杂模板-列表的例子：

```
objects:
- apiVersion: v1
  kind: Secret
  metadata:
    name: ${NAME}
. . . .
- apiVersion: v1
  kind: Service
  metadata:
    name: ${NAME}
  spec:
    ports:
    - name: web
      port: 8080
      targetPort: 8080
    selector:
      name: ${NAME}
- apiVersion: v1
  kind: Route
  metadata:
    name: ${NAME}
  spec:
    host: ${APPLICATION_DOMAIN}
    to:
      kind: Service
      name: ${NAME}
```

技巧和提示

- 使用两个空格来缩进
  - 不要使用制表符
- 使用支持YAML的文本编辑器
  - 最好是使用垂直线来对齐缩进
- 使用在线YAML验证器
- 使用 `export EDITOR=/usr/bin/vim` 来突出显示使用`oc edit`编辑时的语法。
- 使用`oc edit`时，应该会看到元素顺序有变化
  - 因为OpenShift将信息存储在`etcd`数据库中，当从那里提取信息时对其进行排序

8. 总结

- OpenShift 工程
- 应用部署步骤
- 应用健康
- 应用扩缩容
- YAML Manifest

- YAML Manifest 组件
- YAML 基础