腾讯云基于kubernetes的应用编排实践

颜卫@腾讯云

2017.06.30

目录

腾讯云

目录

腾讯云

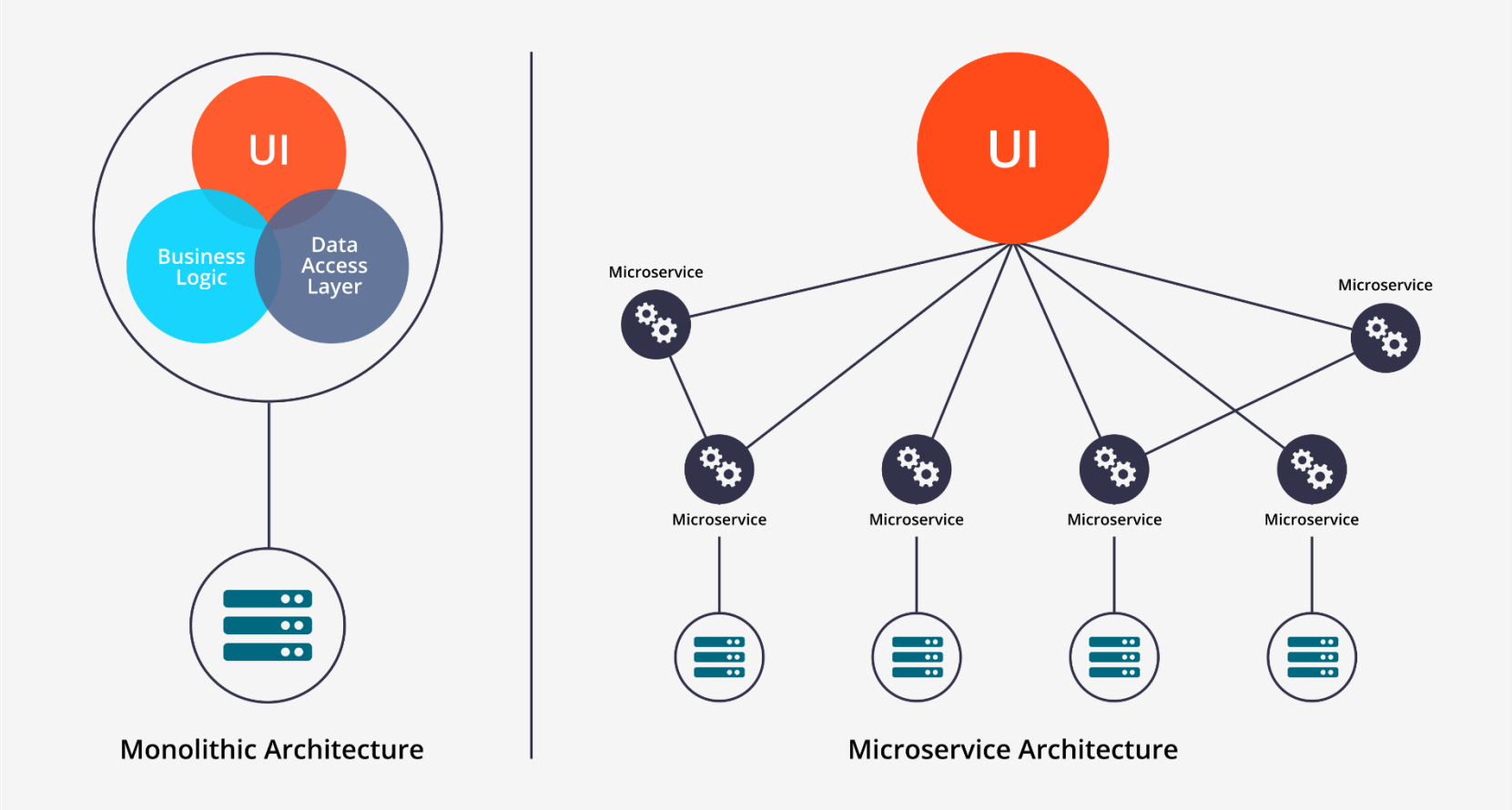
引言--为什么需要应用编排

单体式应用的问题

- 开发效率低
- 维护困难
- 稳定性差
- 扩展性差

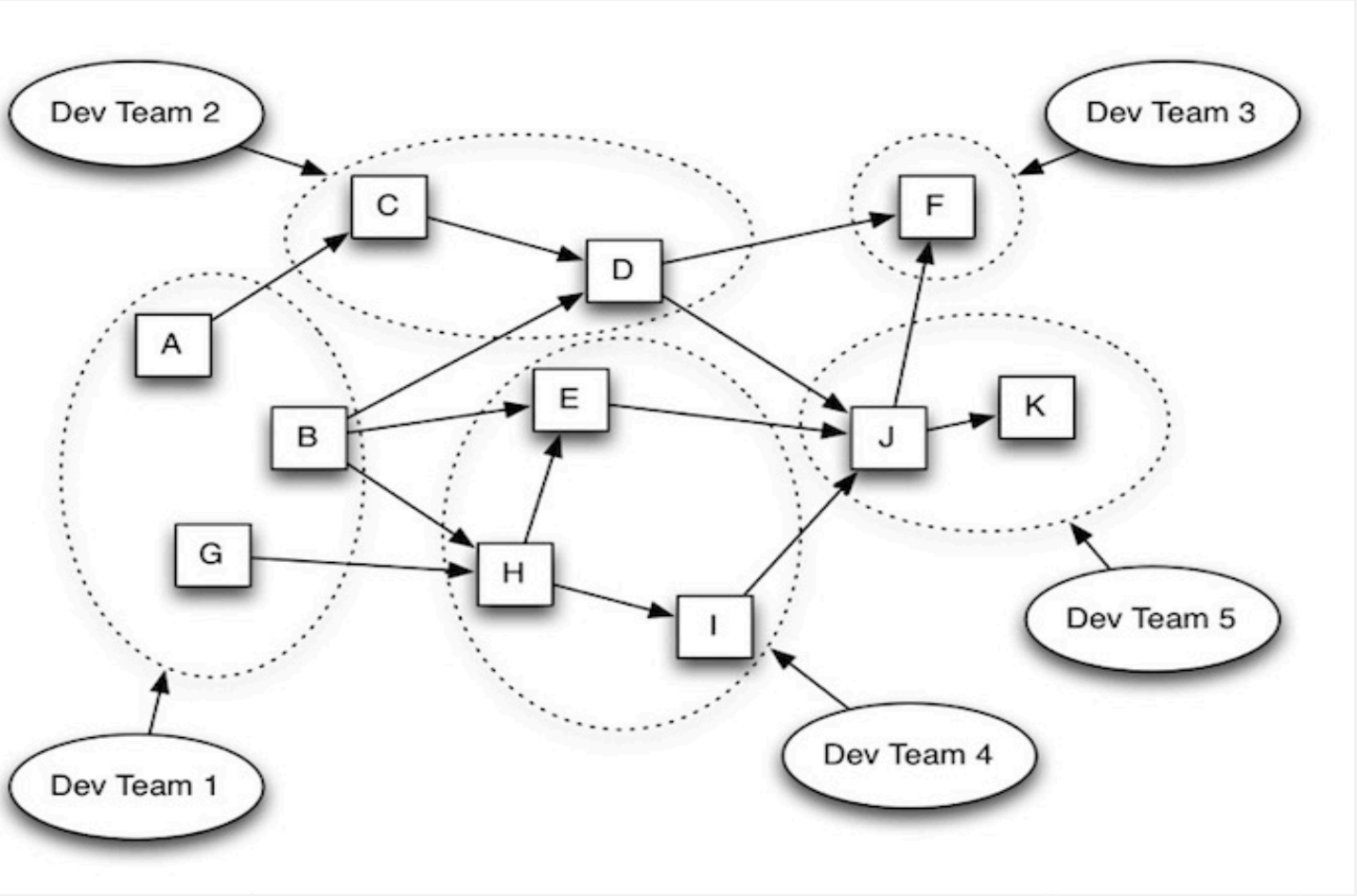
微服务架构带来的收益

- ✓ 开发效率高
- ✓ 维护简单
- ✓ 稳定性好
- ✓ 扩展性好



腾讯云

引用--为什么需要应用编排



随着服务数量的增加：

如何管理服务间的依赖关系？

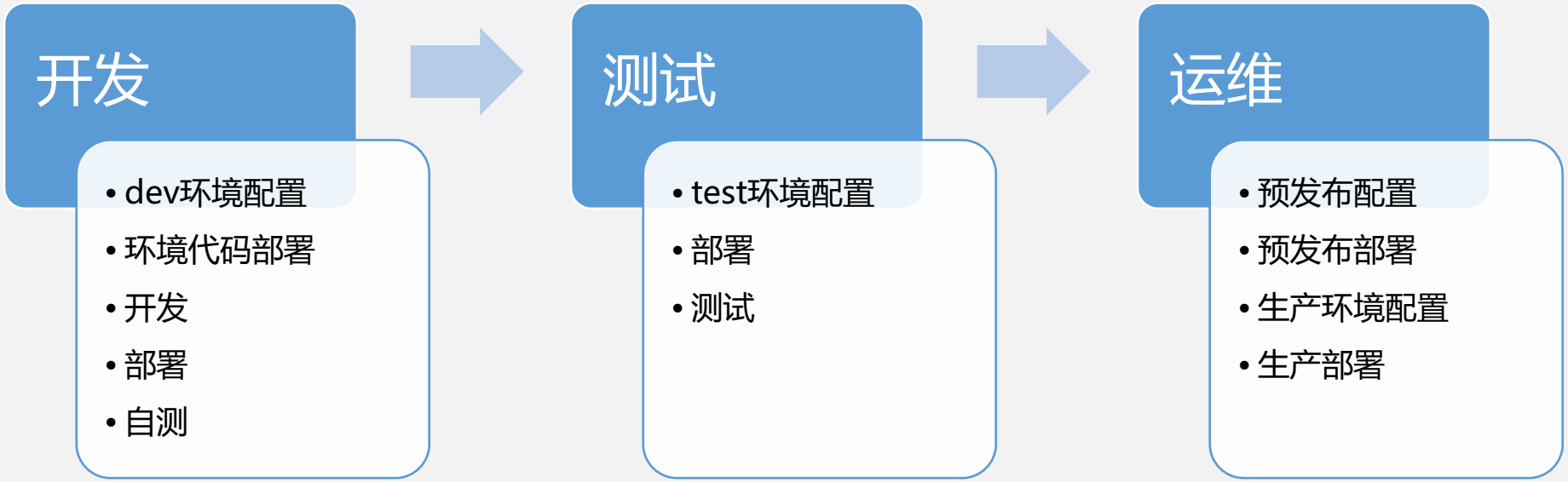
如何处理服务的更新和部署？

微服务部署带来的挑战：

- ✓ 服务数量急剧增多
- ✓ 服务自身配置的管理
- ✓ 服务直接依赖关系的管理
- ✓ 环境信息的管理(不同环境服务有不同的配置)

腾讯云

引用--为什么需要应用编排



开发、测试、预发布、正式环境等 不同环境的部署，增加了服务管理的复杂性。

目录

腾讯云

kubernetes社区应用编排发展现状

kubernetes社区编排方案中，Helm基于Charts包的实现方案占主导地位。
目前Helm已经成为kubernetes下应用编排的唯一子项目。之前推出Helm项目的Deis公司已经被微软收购。

Helm charts包示例：

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  charts/             # OPTIONAL: A directory containing any charts upon which this chart depends.
  templates/          # OPTIONAL: A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

腾讯云

Helm Charts包示例

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "fullname" . }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: {{ template "fullname" . }}
    spec:
      containers:
      - name: {{ template "fullname" . }}
        image: "{{ .Values.image }}"
        imagePullPolicy: {{ default "" .Values.imag
        env:
        - name: ALLOW_EMPTY_PASSWORD
        {{- if .Values.allowEmptyPassword }}
          value: "yes"
        {{- else }}
          value: "no"
        {{- end }}
```

分支语句

内置变量

变量替换

```yaml
{{- if .Values.persistence.enabled -}}
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "fullname" . }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
  annotations:
  {{- if .Values.persistence.storageClass }}
    volume.beta.kubernetes.io/storage-class: {{ .Value
  {{- else }}
    volume.alpha.kubernetes.io/storage-class: default
  {{- end }}
spec:
  accessModes:
    - {{ .Values.persistence.accessMode | quote }}
  resources:
    requests:
      storage: {{ .Values.persistence.size | quote }}
{{- end -}}
```

函数

管道

腾讯云

kubernetes社区应用编排发展现状

kubernetes的问题

- 原生支持通过服务和label进行管理

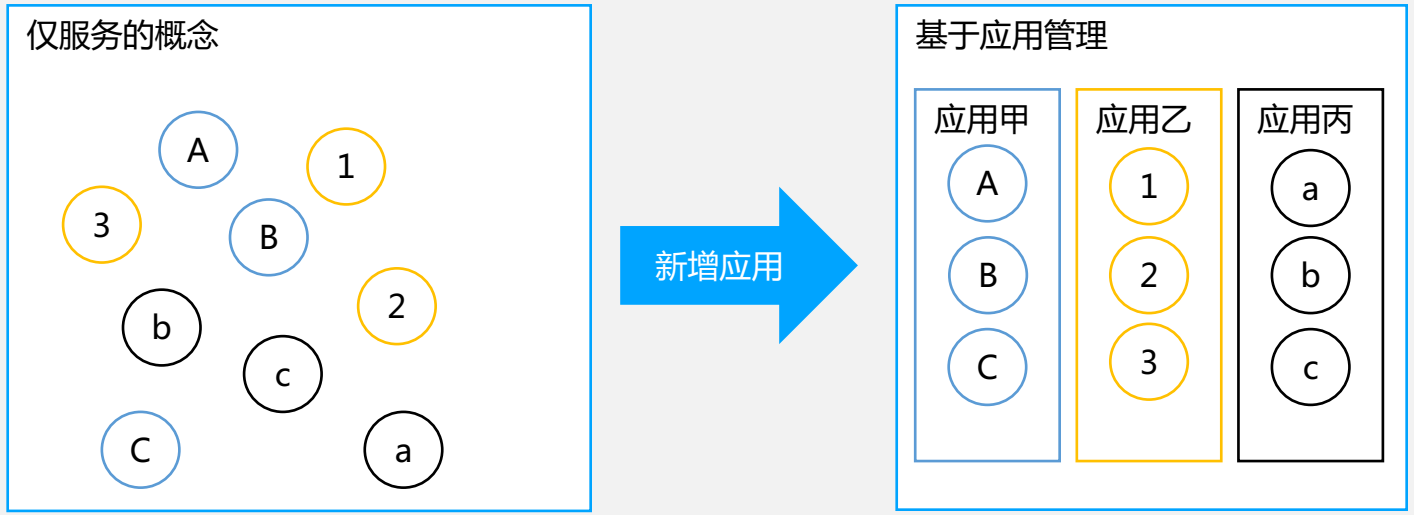
Helm的问题

- 更侧重于包管理
- 语法复杂，学习成本高
- 不支持按照服务更新和管理
- 目前不支持服务启动顺序管理
- 不支持差异化比较

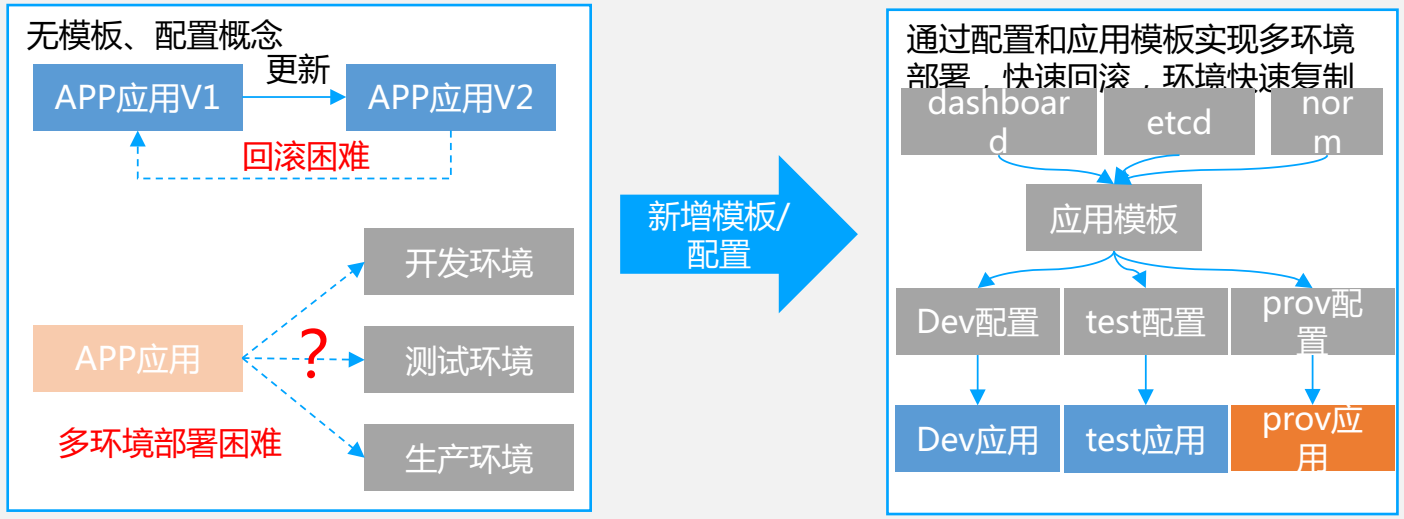
目录

腾讯云

腾讯云容器服务应用编排



应用：包括描述多个服务以及这些服务间的相互调用依赖关系 ，**方便用户管理服务。**

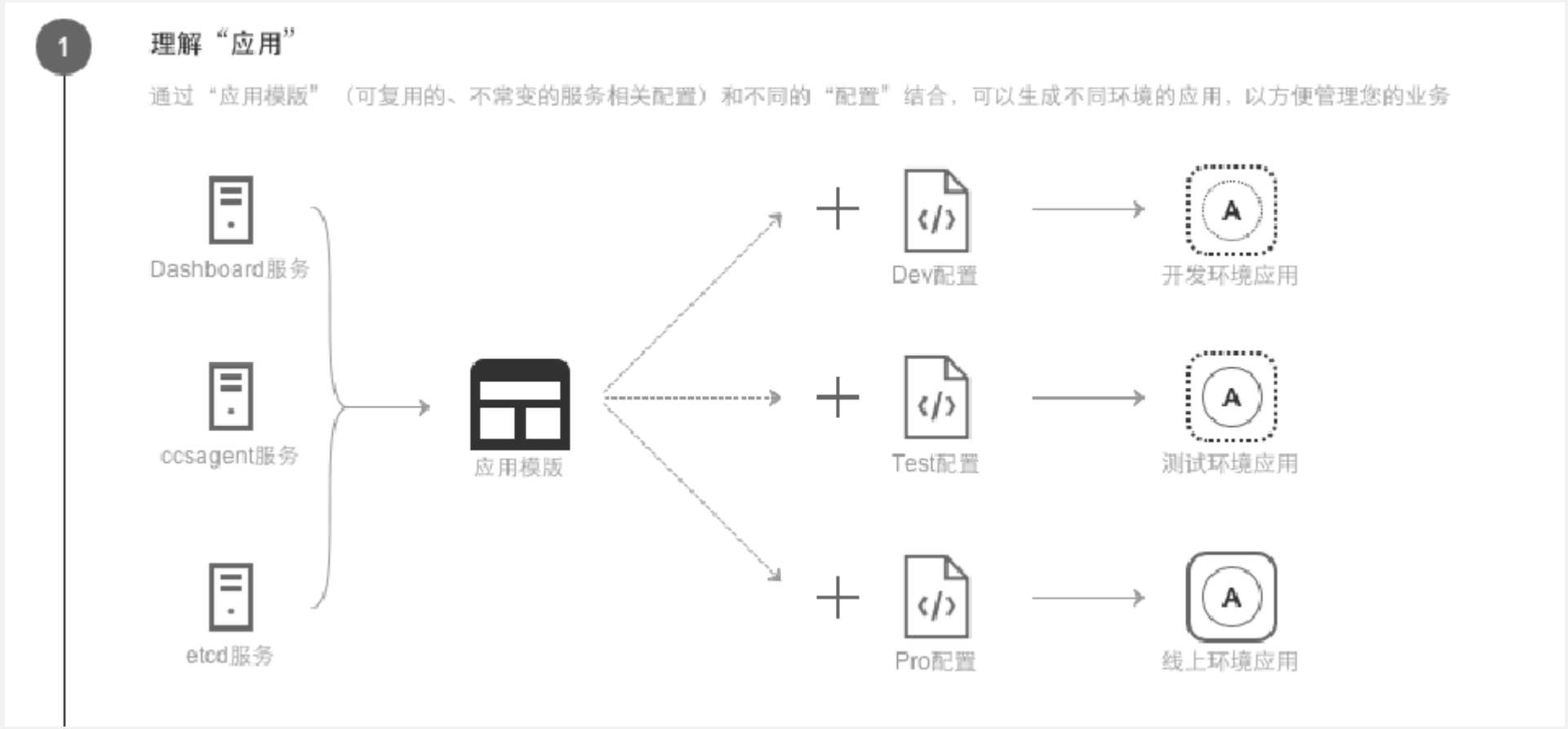


应用模板：包括多个服务的定义加一个默认配置，通过应用模板+配置项的组合，**方便用户部署相同应用的不同环境。**

配置项：将应用中常变的值以变量的形式替代，通过应用模板+配置项的组合，**方便用户部署相同应用的不同环境。**配置项支持多版本，**方便用户进行更新和回滚应用。**

腾讯云

腾讯云容器服务应用编排



应用模板

+

应用配置

=

应用实例

腾讯云

腾讯云容器服务应用编排-配置管理

配置管理的作用：

- 1、支持多环境部署
- 2、支持灵活的服务变更
- 3、服务之间依赖关系管理

配置管理实现：

- a、Helm变量渲染
- b、kubernetes的ConfigMap
- -- 环境变量
- -- 配置文件挂载

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "fullname" . }}
    chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
    release: "{{ .Release.Name }}"
    heritage: "{{ .Release.Service }}"
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: {{ template "fullname" . }}
    spec:
      containers:
      - name: {{ template "fullname" . }}
        image: "{{ .Values.image }}"
        imagePullPolicy: {{ default "" .Values.imag
        env:
        - name: ALLOW_EMPTY_PASSWORD
        {{- if .Values.allowEmptyPassword }}
          value: "yes"
        {{- else }}
          value: "no"
        {{- end }}
```

Helm 基于Go Template实现变量渲染

腾讯云

a. 以环境变量方式引用

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.how
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.type
      envFrom:
        - configMapRef:
            name: env-config
  restartPolicy: Never
```

b. 以参数方式引用

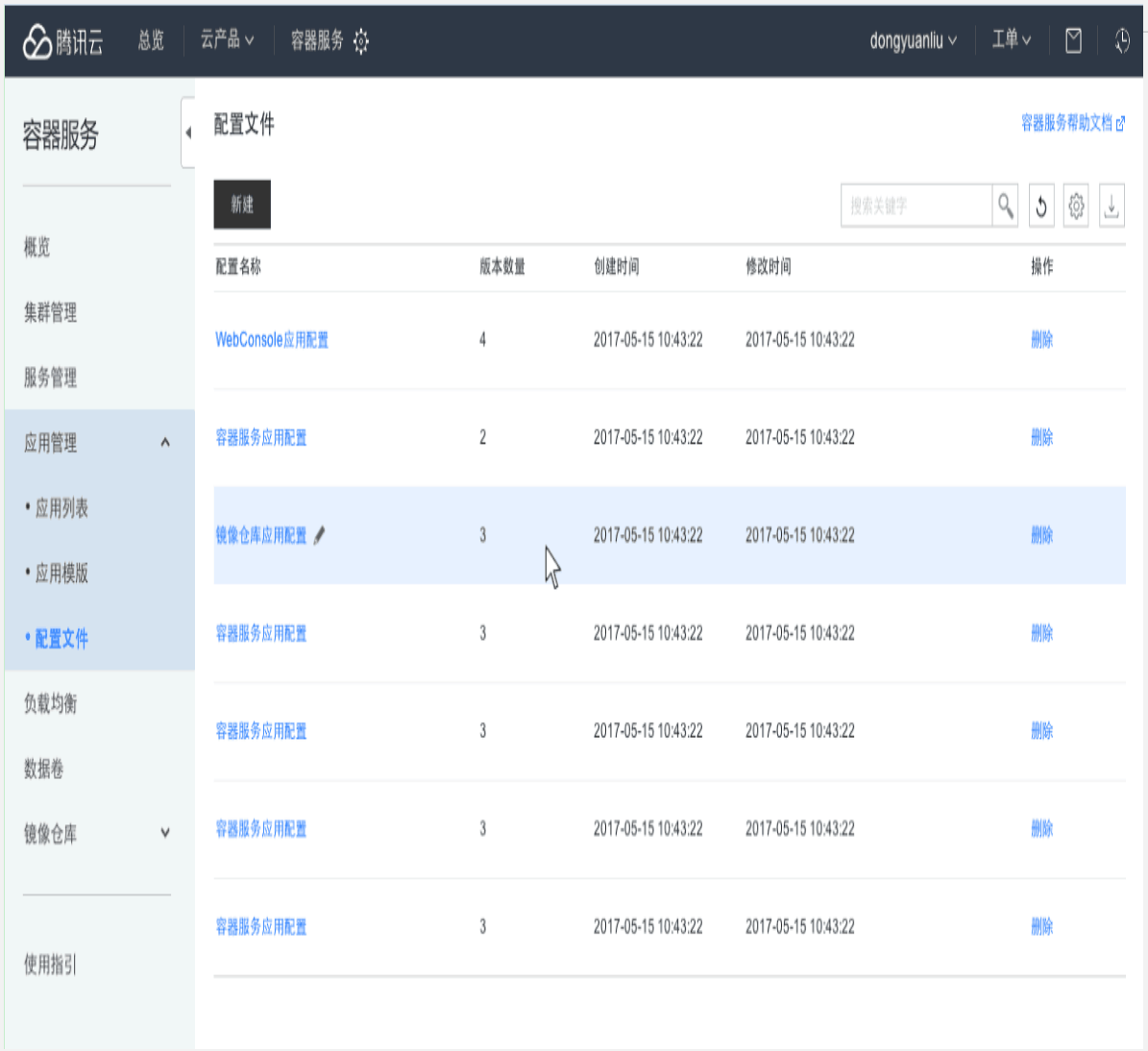
```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY)
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.how
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.type
  restartPolicy: Never
```

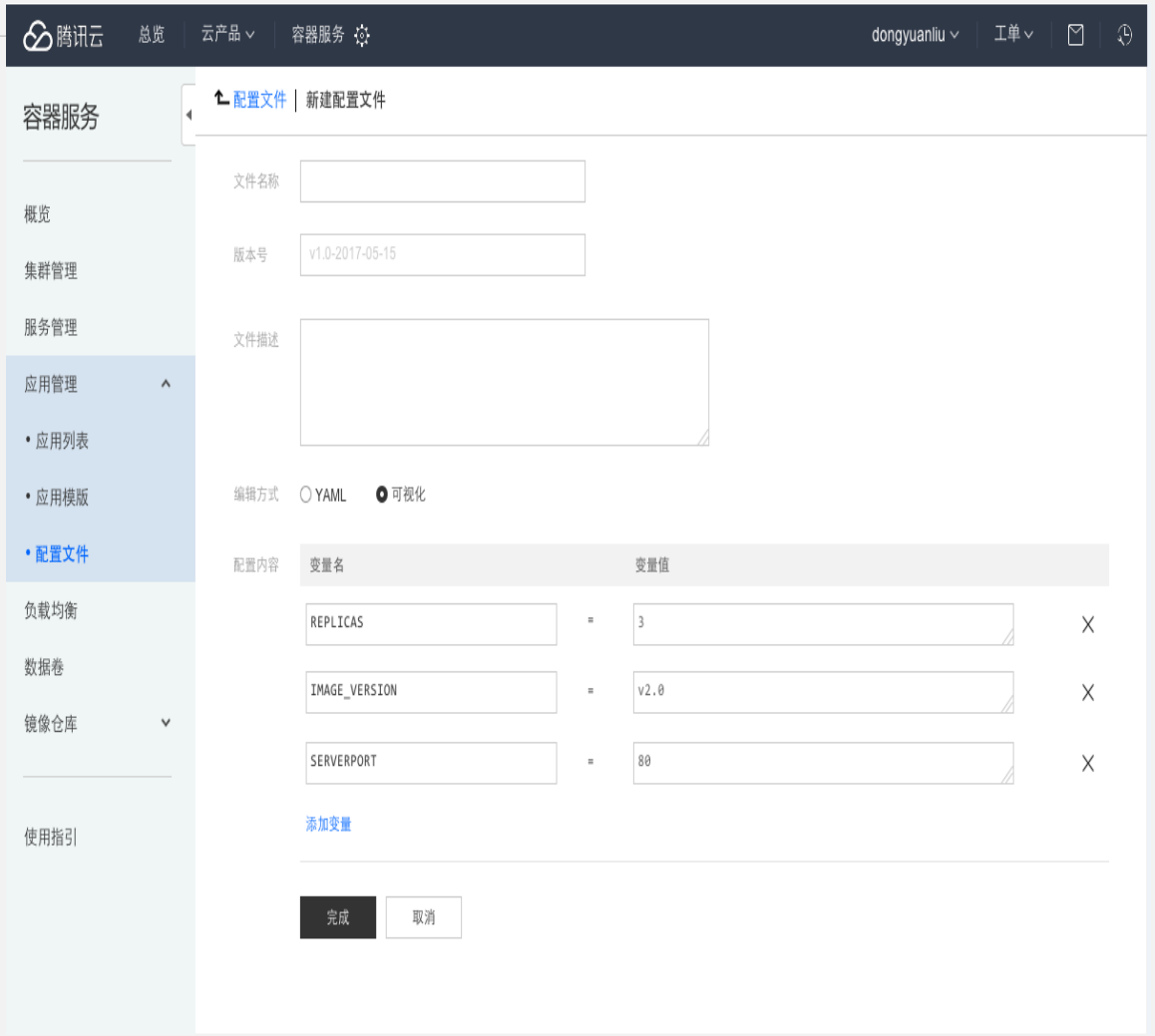
C、直接用ConfigMap的数据填充Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: vol-test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "cat /etc/config/special.how" ]
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```

kubernetes的ConfigMap的3种方式

腾讯云容器服务应用编排-配置管理





腾讯云

腾讯云容器服务应用编排-应用模板

应用模板的主要作用：

✓应用的快速克隆

✓应用的多环境部署

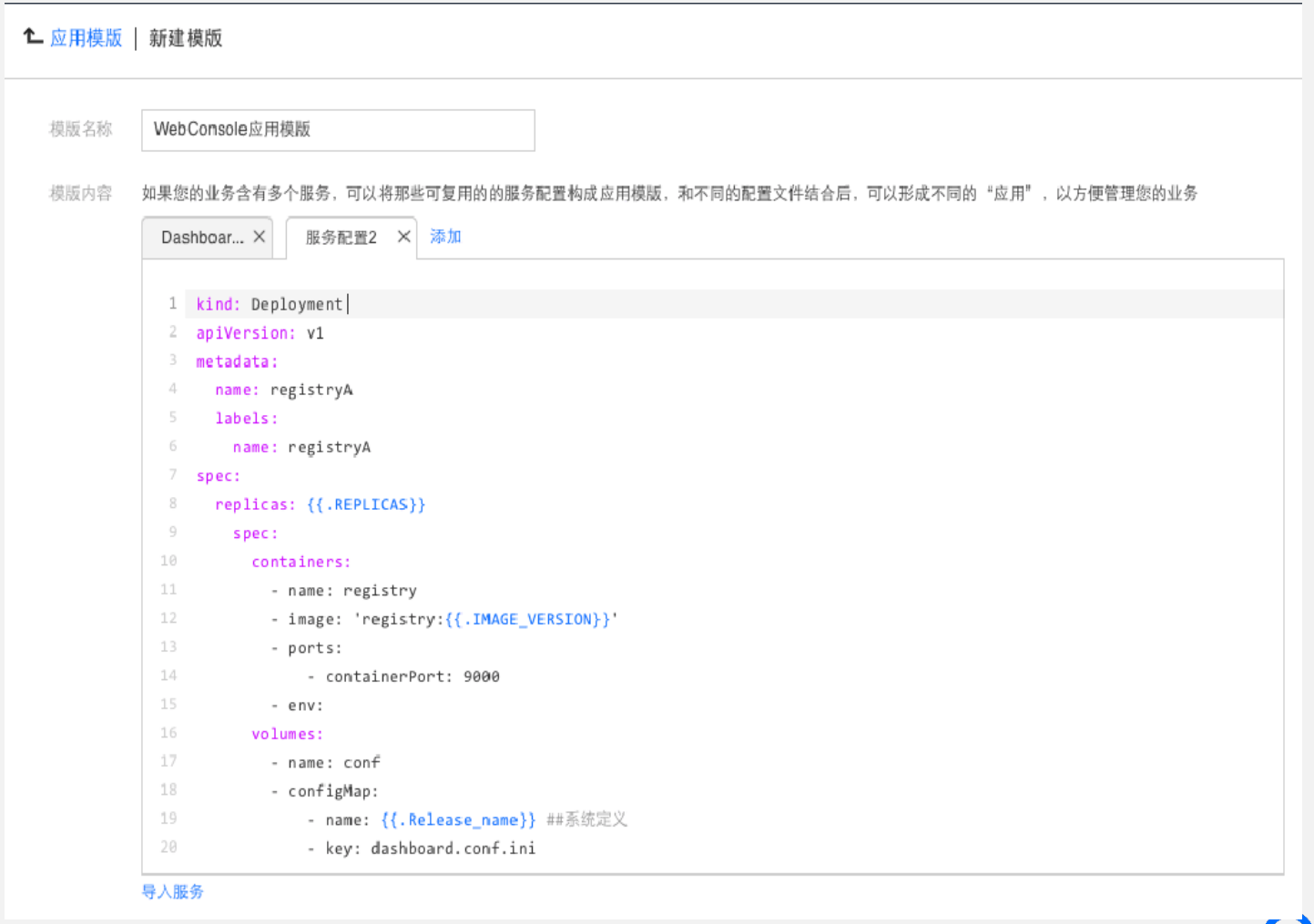
✓应用市场

主要实现方式：

➢应用模板中包含多个服务

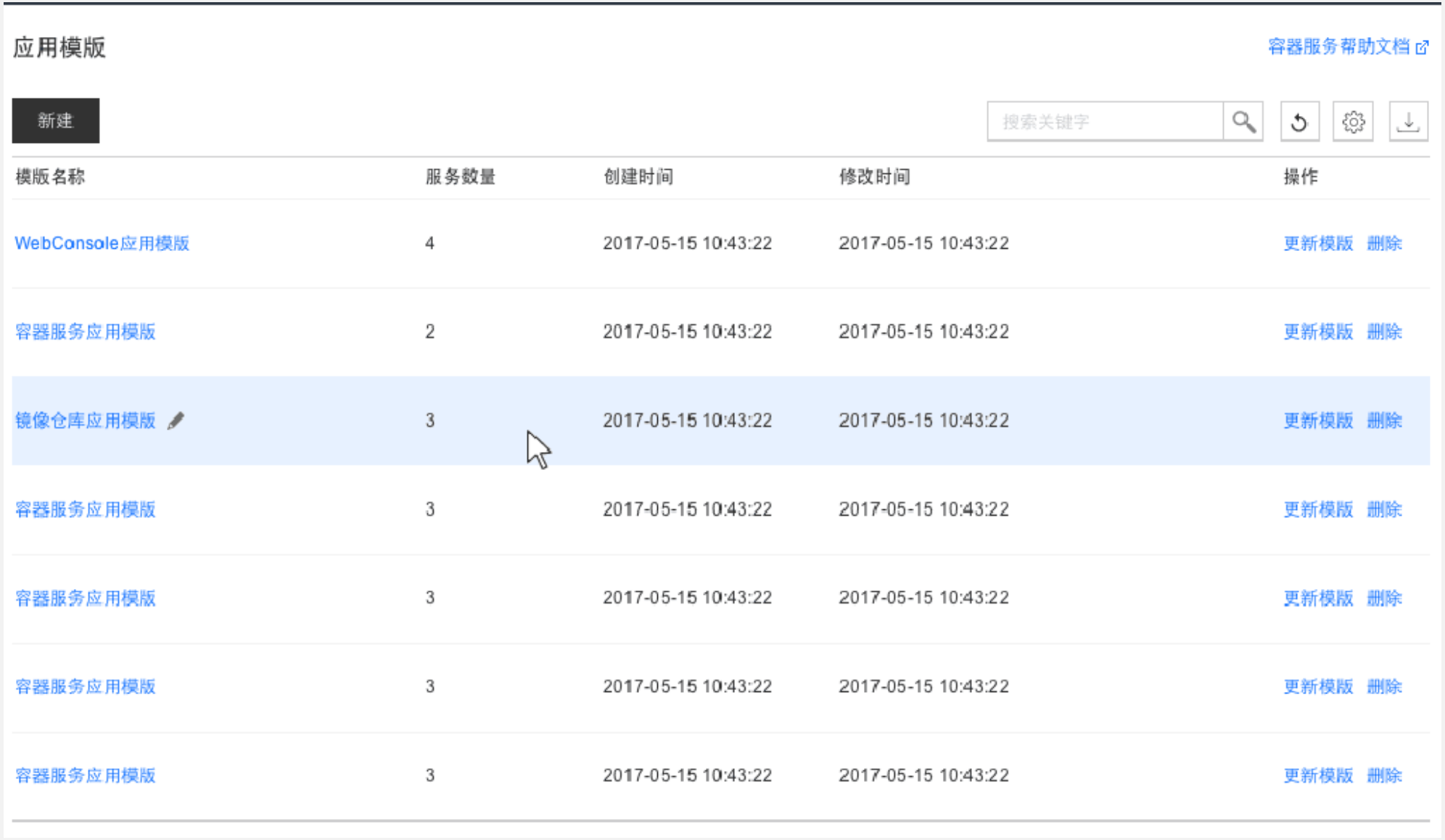
➢基于GoTemplate的信息描述

实现方式与Helm保持一致，便于后期与社区对接



腾讯云

腾讯云容器服务应用编排-应用模板



腾讯云

腾讯云容器服务应用编排-应用管理

应用的主要作用：

基于应用管理多个服务

应用中支持的功能：

✓同一应用服务筛选

✓服务之间的关联管理

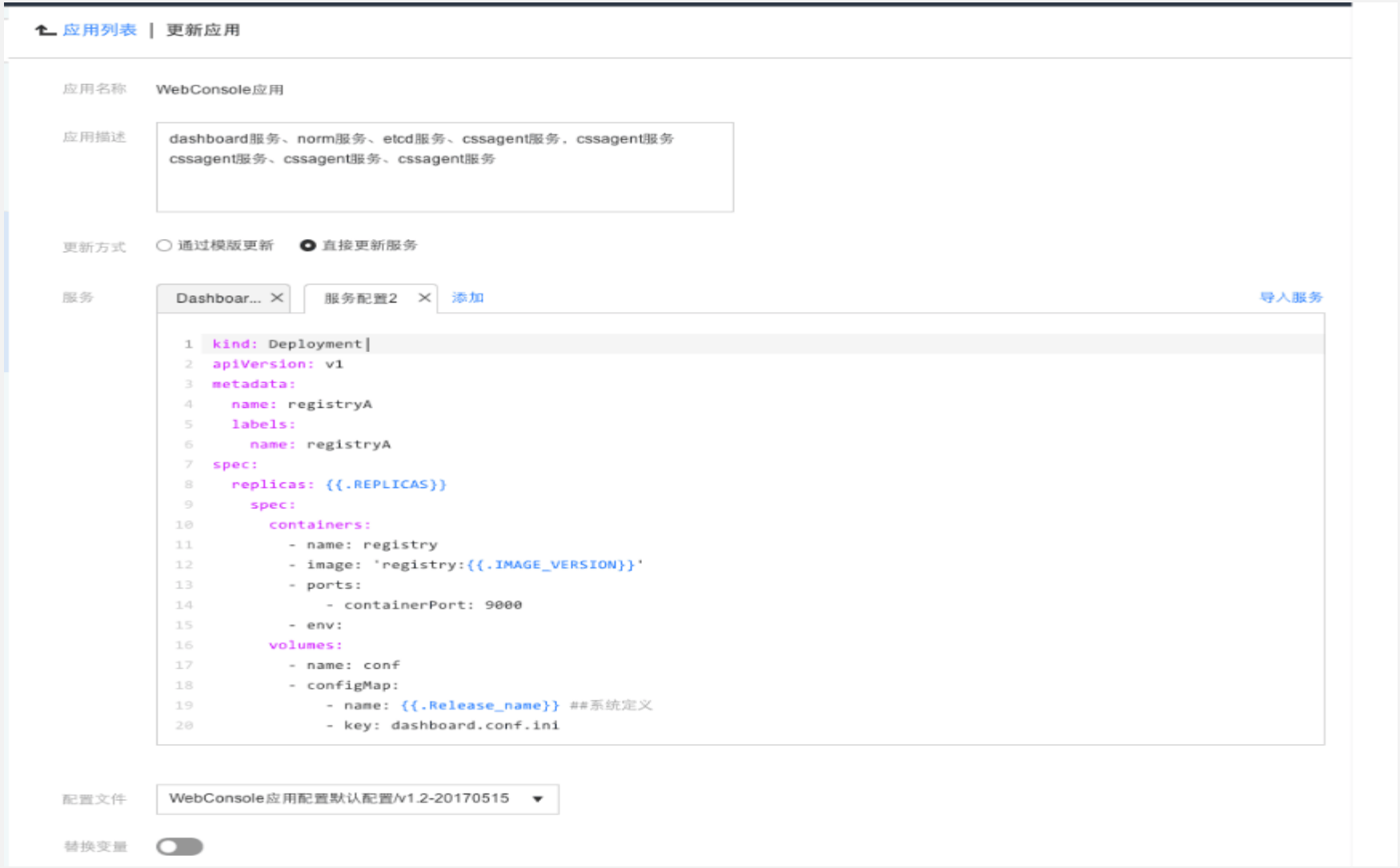
✓服务修改显示修改的状态

腾讯云

腾讯云容器服务应用编排-应用管理

腾讯云

腾讯云容器服务应用编排-应用管理



腾讯云

展望

1. 启动顺序控制
2. 应用下日志聚合
3. 调用关系展示
4. 公共模板与应用市场

Q&A