云原生服务中心

服务提供商指南

文档版本	02
发布日期	2023-08-30

华为云计算技术有限公司

商标声明

HUAWEI和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

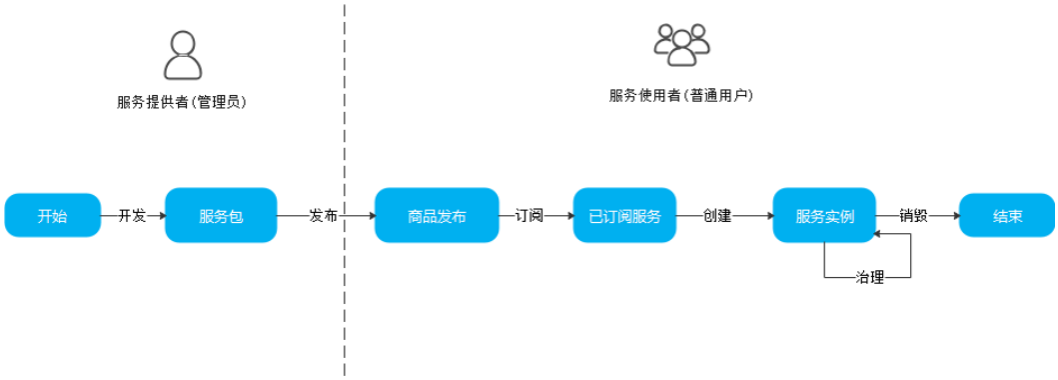
由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 概述

服务提供商是面向OSC合作伙伴，致力于与OSC合作并对外提供商用服务的企业或开发者，本文档介绍提供商如何上传和发布商用服务。OSC为客户提供开箱即用的云原生服务，可以通过OSC部署到任意基础设施，包括华为云CCE集群、第三方集群等场景。服务提供商在发布服务时需要提供服务包，服务包中包含部署需要的配置文件以及镜像，并按照OSC可识别的路径放置，保证服务发布中用户可以一键部署。

图 1-1 商品服务生命周期



2 服务包制作

以etcd operator 0.9.4版本和etcd helm 6.7.0版本为例，介绍如何制作服务包。

2.1 前置检查

- 服务包发布到云市场中，需要提供病毒扫描报告，保证发布上架的服务中不存在任何病毒，并在发布商品时提供给云市场运营人员审核，审核通过后商品方可发布成功。
- 确保Helm模板中的镜像配置都引用values.yaml中的配置，如果不满足，请参见**排查和编辑镜像地址**排查和整改。
- 确保operator包中的镜像配置都引用*.clusterserviceversion.yaml中的配置，如果不满足，请参见**排查和编辑镜像地址**排查和整改。
- 如果服务需要支持多CPU架构部署场景，请确保提供的不同架构镜像都能够使用同一部署包资源描述文件进行部署。

2.2 约束与说明

OSC的服务规范兼容Helm和Operator，提供商无需修改业务代码，只需要打包成满足OSC规范的服务包，就可以发布到OSC上。

想使用OSC提供的高级能力，比如监控、日志等运维能力，在Helm或者Operator服务包中添加配置文件，对接平台运维能力，然后再按照**制作流程**的指导制作服务包。

2.2.1 服务包目录结构约束

OSC作为一个服务生命周期管理平台，有自定义的服务模型，服务需满足目录格式要求才能被OSC解析，发布到OSC上。

```
{ServiceName}-{Version}.zip              #【必选】服务包
└─{ServiceName}/                         #【必选】服务包目录
    ├─package/                           #【必选】部署包目录
    │   ├─{serviceName}-{Version}.zip    #【必选】开源operator部署包或者转换后的osc格式部署包
    │   └─{serviceName}-{Version}.tgz    #【必选】Helm部署包
    └─images/                            #【可选】镜像目录
        ├─{images1}.tar                  #【可选】镜像文件
        ├─{images2}.tar
        ├─ ...
        └─ mapping.yaml                  #【可选】镜像地址替换映射文件
```

须知

package目录只有一个部署包，如果是由Helm改造，就是{serviceName}-{Version}.tgz，如果是Operator改造，是{serviceName}-{Version}.zip，如果是OSC格式转换，是{serviceName}-{Version}.zip。

命名规范：

- serviceName：服务名称。目前服务名称仅接受英文大小写字母、数字及中划线（-）的组合。在服务发布中，创建服务名称其输入长度最大为64个字符。
- version：服务的版本号。服务版本号请遵循SemVer规范进行书写，但不支持SemVer中带有+的版本号。

服务包各个文件命名以及约束

名称	格式	建议参数说明	用途	必选
服务包	{ServiceName}-{Version}.zip	建议服务名+版本号+zip命名	用于发布到OSC市场的最终交付包，zip格式压缩包。	是
服务包目录	{ServiceName}	建议服务名来命名	总服务包目录。	是
部署包目录	package	固定名称，不可修改	OSC兼容两种开源规范，Helm和Operator，package目录用于存放服务的Helm模板或者Operator包，两种类型二选一。	是
osc部署包	{serviceName}-{Version}.zip	服务名+版本号+zip命名	osc部署包，参考《OSC服务开发者指南》章节2.2《OSC服务规范》转换operator或Helm生成的服务包。	是
operator部署包	{serviceName}-{Version}.zip	服务名+版本号+zip命名	operator部署包，参考Operator framework方式生成。	是
helm部署包	{serviceName}-{Version}.tgz	服务名+版本号+tgz命名	Helm部署包，参考开源Helm格式生成即可。	是
镜像目录	images	固定名称，不可修改	存放服务包的镜像目录，当镜像是让用户从外部拉取，镜像目录可不要。	否
镜像文件	{images}.tar	镜像文件，tar格式结尾	镜像文件。	否
镜像地址索引文件	mapping.yaml	固定名称，不可修改	OSC通过这个文件判断Helm模板或者Operator包中配置镜像的字段，以便将这些镜像地址替换成OSC仓库中实际的镜像地址。	否

示例

- etcd Operator服务

```
etcd.zip
  | --- etcd/
      | --- package/
          | --- etcd-operator-0.9.4.zip        #etcd Operator包，必须是zip格式的压缩包
      | --- images/
          | --- etcd-operator-0.9.4.tar        #etcd Operator的镜像
          | --- etcd-3.5.0.tar                 #etcd 的镜像
          | --- mapping.yaml
```

- etcd Helm服务

```
etcd.zip
  | --- etcd/
      | --- package/
          | --- etcd-helm-6.7.0.tgz            #etcd helm模板包，必须为tgz的格式
      | --- images/
           | --- etcd-3.5.0.tar             # etcd 的镜像
           | --- mapping.yaml
```

2.2.2 服务包镜像名称约束

镜像目录存放部署目录中需要使用到的镜像文件，镜像文件以tar格式存储。镜像文件命名规范只允许包含大小写字母、数字、下划线、中划线及点，且不能以下划线、中划线及点结尾。

images目录下存放服务运行所需的容器镜像，根据服务实际情况确定支持单CPU架构还是多CPU架构：

- 单CPU架构：一般制作生成的容器镜像，其默认只能在某一种CPU架构的节点上运行,比如X86或ARM。
- 多CPU架构：由多个单架构容器镜像制作而成，能够根据节点架构类型而选择对应的架构镜像并运行，多架构镜像制作方法可以参考**CCE中使用x86和ARM双架构镜像**。

如果服务仅支持一种CPU架构，则直接提供对应CPU架构的镜像即可；如果支持多CPU架构，则需要同时提供适配不同CPU架构的镜像，由OSC统一转换存储，无须用户自己制作。

2.2.3 服务内容说明

服务商用发布版本上架到OSC云原生市场后，会读取服务包的部分描述信息并展示在服务包详情中，因此需要用户在服务包中填写必要的信息，从而保证详情页面展示正常。

根据服务包格式类型，需补充说明内容如下：

- Helm Chart服务包

  Chart服务包其主体描述信息存储在Chart.yaml文件中，其中Chart.yaml文件中有如下几个必填字段内容：

```
version: 1.0.0
name: example-helm
apiVersion: v1
description: A Helm chart for example.
maintainers:
  - name: huawei
    email：osc@huawei.com
annotations:
  source: ISV
  categories: Database
```

```
architecture: x86_64,aarch64
scenes: CCE     # 华为云
```

说明

categories表示此服务包所属类别，当前支持的有如下几种：

- AI/Machine Learning： AI/机器学习。
- Application Runtime: 运行时。
- Big Data: 大数据。
- Database： 数据库。
- Monitoring：监控。
- Security：安全。
- Streaming & Messaging：流媒体&消息。
- Integration & Delivery：集成交付。
- Logging & Tracing： 日志。
- Developer Tools：开发工具。
- Networking：网络。
- Others：其他类型服务；如服务类型非11种，均为其他类型服务。

- Operator-Framework Operator服务包

  Operator-Framework服务中的描述信息存放在*.clusterserviceversion.yaml中，其中必填项如下：

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
  annotations:
    alm-examples： {xxx}  # json结构体，用于描述文件的cr内容
    categories： Big Data
    description： An example for operator
    scenes: UCS,CCE
    source: ISV
    architecture: aarch64
spec:
  maintainers:
  - email: osc@huawei.com
    name: osc huawei
  provider:
    email: osc@huawei.com
    name: osc huawei
```

- OSC服务格式包

OSC服务包格式中详情来源于metadata.yaml文件，其中必填项如下：

```
name: example-operator
version: "1.0.0"
appVersion: 2.1.2
displayName: Example osc
briefDescription: example osc with an example instance and action
source: ISV
architecture:
  - x86_64
  - aarch64
categories:
  - Database
maintainers:
  - email: test@test.com
    name: test
provider:
  name: Example provider
  url: https://example.com/
```

```
scenes:
  - CCE
```

2.2.4 镜像拉取配置说明

此配置主要针对用户发布的服务包中带有镜像，需要授权给对应的购买用户者下载使用时需要增加的步骤。

用户需要在用户部署的deployment.yaml、statefulset.yaml等包含镜像下载的文件中增加如下配置。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example
spec：
  replicas: 1
  spec:
    serviceAccountName: details
    containers:
    - name: details
      image: "{{ .Values.global.hub }}/details:1.5.0"
      imagePullPolicy: IfNotPresent
    restartPolicy: Always
    imagePullSecrets:
      - name: default-secret
```

2.3 制作流程

2.3.1 制作服务包目录

新建一个文件夹，可任意命名，以etcd服务为例，暂定命名为etcd，在etcd文件夹下创建两个子目录，分别命名为package，images。

```
etcd
    | --- package
    | --- images
```

2.3.2 放置镜像到 images 目录

images目录下存放服务部署时需要的所有镜像。以etcd为例，etcd服务包含x86和ARM两种CPU架构的镜像，分别叫etcd-3.5.0-x86_64.tar，etcd-3.5.0-aarch64.tar，那么目录结构为：

```
etcd/
    | --- package/
    | --- images/
        | --- etcd-3.5.0-x86_64.tar
        | --- etcd-3.5.0-aarch64.tar
```

有的服务比较复杂，依赖其它的服务，比如kafka依赖zookeeper，zookeeper的镜像也要放置在images目录下：

```
kafka/
    | --- package/
    | --- images/
        | --- kafka-2.5.0-x86_64.tar
        | --- kafka-2.5.0-aarch64.tar
        | --- zookeeper-3.4.0-x86_64.tar
        | --- zookeeper-3.4.0-aarch64.tar
```

2.3.3 打包到 package 目录

Helm模板或者operator包有两个修改点：

- 标记服务的来源，即是来源于开源、华为或者生态伙伴；
- 统一镜像地址，OSC格式包引用lifecycle.yaml及csd.yaml中配置，Helm模板引用values.yaml中的配置，operator引用*.clusterserviceversion.yaml中的配置。

须知

提供商上传服务包后，镜像会保存到OSC的仓库中，提供商在上传前无法获取镜像的实际保存地址，OSC服务会将value.yaml或者*.clusterserviceversion.yaml中默认的镜像地址替换成实际的镜像地址，其它地方的配置不予替换，因此如果其它地方没有引用lifecycle.yaml、csd.yaml、values.yaml或者*.clusterserviceversion.yaml中的配置，在部署服务实例时，拉取镜像会失败。

标识服务来源

OSC提供来自开源、华为自研以及生态伙伴的服务，生态伙伴需要在服务包中固定字段进行标记说明服务来自生态伙伴。

- operator服务

  以etcd operator为例，etcdoperator.v0.9.4.clusterserviceversion.yaml的内容为：

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
  annotations:
    capabilities: Full Lifecycle
    categories: Database
    containerImage: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
    createdAt: 2019-02-28 01:03:00
    description: Create and maintain highly-available etcd clusters on Kubernetes
    repository: https://github.com/coreos/etcd-operator
    tectonic-visibility: ocs
  name: etcdoperator.v0.9.4
  namespace: placeholder
...
```

步骤1 编辑文件。

在metadata/annotations添加source字段，取值为ISV。

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
  annotations:
    source: ISV
    capabilities: Full Lifecycle
    categories: Database
    containerImage: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
    createdAt: 2019-02-28 01:03:00
    description: Create and maintain highly-available etcd clusters on Kubernetes
    repository: https://github.com/coreos/etcd-operator
    tectonic-visibility: ocs
  name: etcdoperator.v0.9.4
  namespace: placeholder
...
```

步骤2 打包到package目录。

把operator包压缩成zip格式，放至package目录下。

注意

package目录下只能包含一个Operator压缩包，确保使用这个包可以将整个服务部署起来。

```
etcd/
  | --- package/
      | --- etcd-operator-0.9.4.zip
  | --- images/
      | --- etcd-3.5.0-x86_64.tar
      | --- etcd-3.5.0-aarch64.tar
  | --- extends/
```

----结束

- Helm服务

  以etcd helm为例，Chart.yaml的内容为：

```
annotations:
  category: Database
apiVersion: v2
appVersion: 3.4.14
dependencies:
  - name: common
    repository: https://charts.bitnami.com/bitnami
    tags:
      - bitnami-common
    version: 1.x.x
description: etcd is a distributed key value store that provides a reliable way to store data across a
cluster of machines
...
```

步骤1 编辑Chart.yaml文件。

如果有annotations属性，则添加子属性source，取值为ISV，如果没有annotations属性，则先添加annotations属性，再添加子属性source。

```
annotations:
  source: ISV
  category: Database
apiVersion: v2
appVersion: 3.4.14
dependencies:
  - name: common
    repository: https://charts.bitnami.com/bitnami
    tags:
      - bitnami-common
    version: 1.x.x
description: etcd is a distributed key value store that provides a reliable way to store data across a cluster of
machines
...
```

注意

package目录下只能包含一个Helm模板包，确保使用这个包可以将整个服务部署起来。

步骤2 打包放置到package目录：

```
etcd/
  | --- package/
      | --- etcd-helm-6.7.0.tgz
  | --- images/
      | --- etcd-3.5.0-x86_64.tar
      | --- etcd-3.5.0-aarch64.tar
      | --- mapping.yaml
```

----结束

排查和编辑镜像地址

提供商上传服务包后，镜像会保存到OSC的仓库中，提供商在上传前无法知道镜像的实际保存地址，OSC在提供商上传服务后只把value.yaml或者*.clusterserviceversion.yaml中配置的镜像地址替换成实际的镜像地址。

须知

提供商需排查Helm模板或者Operator服务包中的其它镜像地址都是引用values.yaml或者*.clusterserviceversion.yaml中的配置。如果满足，可直接查看**替换镜像地址配置**。

以etcd helm为例，Helm模板目录结构如下所示：

```
etcd/
  | --- templates/
      | --- secrets.yaml
      | --- servicemonitor.yaml
      | --- snapshot-pvc.yaml
      | --- statefulset.yaml
      | --- svc-headless.yaml
      | --- svc.yaml
  | --- Chart.lock
  | --- Chart.yaml
  | --- README.md
  | --- values.yaml
```

其中values.yaml文件内容是：

```
image:
  registry: docker.io
  repository: bitnami/etcd
  tag: 3.4.14-debian-10-r44
  debug: false
volumePermissions:
  enabled: false
  image:
    registry: docker.io
    repository: bitnami/minideb
    tag: buster
    pullPolicy: Always
  resources:
    limits: {}
    #   cpu: 100m
    #   memory: 128Mi
    requests: {}
    #   cpu: 100m
    #   memory: 128Mi
...
```

templates/statefulset.yaml中引用了values.yaml配置镜像地址的变量：

```
containers:
  - name: etcd-snapshotter
    image: {{ include "etcd.image" . }}
    imagePullPolicy: {{ .Values.image.pullPolicy | quote }}
```

服务包上传后实际的镜像地址是swr.cn-east-3.myhuaweicloud.com/osc-opensource/etcd:3.4.14-debian-10-r44，OSC会自动替换服务包中values.yaml中的镜像地址，保证服务部署时能从OSC的仓库中拉取镜像，values.yaml会被替换成：

```
image:
  registry: swr.cn-east-3.myhuaweicloud.com
  repository: osc-opensource/etcd
  tag: 3.4.14-debian-10-r44
```

因为Helm模板中其它镜像地址是引用values.yaml的配置，因此OSC替换了values.yaml后能保证部署时拉取到正确的镜像。

说明

对于Operator服务，请排查服务包中的镜像地址都是引用*.clusterserviceversion.yaml中的配置。

配置实例版本定义信息(可选)

OSC提供配置实例版本定义信息以支持实例升级的能力。

- Operator类型实例

  以redis为例， redis的cr内容如下所示：

```
apiVersion: redis.osc/v1
kind: Redis
metadata:
  annotations:
    osc.huawei.com/package-source: public
    osc.io/occupied-port: 135,139
  creationTimestamp: '2021-09-29T03:21:57Z'
  finalizers:
    - storage.finalizers.redis.cluster
  generation: 2
  name: redis-fwpydh
  namespace: default
  resourceVersion: '91743432'
  selfLink: /apis/redis.osc/v1/namespaces/default/redises/redis-fwpydh
  uid: 96f0203c-0ae0-48bb-b2b8-b08d2055b0e2
spec:
  config:
    name: default-redis-fwpydh-unvu7g
    properties:
      aof-load-truncated: 'yes'
      aof-use-rdb-preamble: 'no'
      appendfsync: everysec
      appendonly: 'no'
      hash-max-ziplist-entries: 512
      hash-max-ziplist-value: 64
      latency-monitor-threshold: 100
      list-max-ziplist-size: -2
      loglevel: notice
      maxauthfailtimes: 100
      maxclients: 10000
      maxmemory-policy: noeviction
      repl-diskless-sync: 'yes'
      set-max-intset-entries: 512
      slowlog-log-slower-than: 10000
      stop-writes-on-bgsave-error: 'no'
      timeout: 0
      zset-max-ziplist-entries: 128
      zset-max-ziplist-value: 64
```

```
image: swr.cn-east-3.myhuaweicloud.com/osc-official/redis:21.9.18_20210918221431
masterSize: 1
mode: RedisHA
......
......
此处中间省略
......
......
phase: Available
serviceAddr: redis-ha-redis-fwpydh.default.svc.cluster.local:6379
serviceAddrReadonly: redis-ha-redis-fwpydh-readonly.default.svc.cluster.local:6379
version: 21.9.18_20210918221431
```

步骤1 编辑文件

实例版本定义配置是在csd文件中配置versionDefinition，实例版本定义支持operator类型和Helm类型。

```
versionDefinition:
  mode: url
  path: spec.image
  tags:
    - 0.0.1
    - 0.0.2
    - 0.0.3
```

步骤2 打包到package目录。

把operator包压缩成zip格式的压缩包，放到package目录下。

----结束

- Helm服务

  以clickhouse helm为例，values.yaml的内容如下：

```
## Timezone
timezone: "Asia/Shanghai"

## Cluster domain
clusterDomain: "cluster.local"

##
## Clickhouse Node selectors and tolerations for pod assignment
## ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#nodeselector
## ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#taints-and-tolerations-beta-feature
##
# nodeSelector: {"beta.kubernetes.io/arch": "amd64"}
# tolerations: []
## Clickhouse pod/node affinity/anti-affinity
##
#affinity:
#  nodeAffinity:
#    requiredDuringSchedulingIgnoredDuringExecution:
#      nodeSelectorTerms:
#      - matchExpressions:
#        - key: "application/clickhouse"
#          operator: In
#          values:
#          - "true"

clickhouse:
  ## StatefulSet controller supports relax its ordering guarantees while preserving its uniqueness and identity guarantees. There are two valid pod management policies: OrderedReady and Parallel
  ## ref: https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/#pod-management-policy
  ##
```

```
  podManagementPolicy: "Parallel"

  ## StatefulSet controller supports automated updates. There are two valid update strategies:
RollingUpdate and OnDelete
  ## ref: https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/#updating-
statefulsets
  ##
  updateStrategy: "RollingUpdate"

  ## Partition update strategy
  ## https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/#partitions
  ##
  # rollingUpdatePartition:

  ##
  ## The path to the directory containing data.
  ## Default value: /var/lib/clickhouse
  path: "/var/lib/clickhouse"
  ##
  ## The port for connecting to the server over HTTP
  http_port: "8123"
  ##
  ## Port for communicating with clients over the TCP protocol.
  tcp_port: "9000"
  ##
  ## Port for exchanging data between ClickHouse servers.
  interserver_http_port: "9009"
  ##
  ## The instance number of Clickhouse
  replicas: "3"
  ## Clickhouse image configuration.
  image: "swr.cn-east-3.myhuaweicloud.com/osctest/clickhouse-server"
  imageVersion: "0.0.1"
  imagePullPolicy: "IfNotPresent"
  imageBusybox: "swr.cn-north-7.myhuaweicloud.com/osctest/busybox:1.26.2"
  #imagePullSecrets:
  ## Periodic probe of container liveness. Container will be restarted if the probe fails. Cannot be
updated.
  ## More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes
  livenessProbe:
    enabled: true
    initialDelaySeconds: "30"
    periodSeconds: "30"
    timeoutSeconds: "5"
    failureThreshold: "3"
    successThreshold: "1"
  ## Periodic probe of container service readiness. Container will be removed from service endpoints if
the probe fails. Cannot be updated.
  ## More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes
  readinessProbe:
    enabled: true
    initialDelaySeconds: "30"
    periodSeconds: "30"
    timeoutSeconds: "5"
    failureThreshold: "3"
    successThreshold: "1"
  ## volumeClaimTemplates is a list of claims that pods are allowed to reference.
  ## The StatefulSet controller is responsible for mapping network identities to claims in a way that
maintains the identity of a pod.
  ## Every claim in this list must have at least one matching (by name) volumeMount in one
container in the template.
  ## A claim in this list takes precedence over any volumes in the template, with the same name.
  persistentVolumeClaim:
    enabled: false
    ## Clickhouse data volume
    dataPersistentVolume:
      enabled: false
      accessModes:
      - "ReadWriteOnce"
```

```
    storageClassName: "-"
    storage: "500Gi"
  ## Clickhouse logs volume
  logsPersistentVolume:
    enabled: false
    accessModes:
    - "ReadWriteOnce"
    storageClassName: "csi-disk"
    storage: "50Gi"
##
## An API object that manages external access to the services in a cluster, typically HTTP.
## Ingress can provide load balancing, SSL termination and name-based virtual hosting.
ingress:
  enabled: false
#   host: "clickhouse.domain.com"
#   path: "/"
#   tls:
#     enabled: false
#     hosts:
#     - "clickhouse.domain.com"
#     - "clickhouse.domain1.com"
#     secretName: "clickhouse-secret"
##
## Clickhouse config.xml and metrica.xml
......
......
后续省略
......
......
```

步骤1 编辑文件。

实例版本定义配置是在csd文件中配置versionDefinition，实例版本定义支持operator类型和Helm类型。

```
versionDefinition:
  mode: tag
  path: clickhouse.imageVersion
  tags:
    - 0.0.1
    - 0.0.2
    - 0.0.3
```

步骤2 打包到package目录。

把Helm包压缩成zip格式的压缩包，放到package目录下。

----结束

2.3.4 替换镜像地址配置

OSC需替换values.yaml或者*.clusterserviceversion.yaml中的镜像地址，mapping.yaml文件确定OSC在values.yaml或者*.clusterserviceversion.yaml中哪些字段用于配置镜像地址。mapping.yaml有多种不同的字段替换方式，镜像根据实际情况选择对应字段，会将镜像分隔成不同内容回填到配置文件中。

```
instance:                                   # helm文件镜像替换描述
- image: xxxxx                                # 定义推送到仓库后的镜像名称，名称必须符合docker镜像命名规范
  tag: xxx                                  # 定义推送到仓库后的镜像版本tag
  address:                                  # 定义value.yaml中用于替换到目标key值的路径以及替换的镜像结果内
容
    fullAddress: xxxx                         # 替换内容1，xxxx指代的value.yaml文件中最外层到替换镜像值的全
路径，以点号相连
    repository: xxx                         # 替换内容2
    prefixAddress: xxx                        # 替换内容3
    repo: xxx                               # 替换内容4
    endpoint: xxx                           # 替换内容5
```

```
    tag: xxx                          # 替换内容6
  package:                             # 定义替换上面address字段中的镜像路径是来自images目录中的镜像
tar包
    x86_64: xxx.tar                     # 声明使用x86格式的镜像tar包替换
    aarch64: xxx.tar                     # 声明使用arm格式的镜像tar包替换
```

对于一个服务，如果是单CPU架构，则只需要配置x86_64或者aarch64之一即可，如果是多CPU架构，则x86_64和aarch64都需配置，osc服务会将两个服务制作成共架构服务包并将最终镜像地址回填到部署文件中。以下分别通过一个Helm服务和Operator服务举例说明，以镜像swr.cn-east-3.myhuaweicloud.com/osc-opensource/redis:0.0.1为例，address中的字段含义如下所示：

swr.cn-east-3.myhuaweicloud.com/osc-opensource/redis:0.0.1

- fullAddress：swr.cn-east-3.myhuaweicloud.com/osc-opensource/redis:0.0.1
- repository：swr.cn-east-3.myhuaweicloud.com/osc-opensource/redis
- prefixAddress：swr.cn-east-3.myhuaweicloud.com/osc-opensource
- repo: osc-opensource/redis
- endpoint：swr.cn-east-3.myhuaweicloud.com
- tag：0.0.1

说明

配置对应值，OSC服务会将推送后的镜像拉取地址对应回填到部署文件中的全量路径key对应的地址中去。

Helm 服务示例

以exampleservice helm服务为例，该服务依赖kafka、redis、zookeeper、mariadb，镜像分别 kafka-2.5.0.tar，redis-6.0.10.tar，zookeeper-3.4.14.tar，mariadb-10.5.8.tar，exampleservice包含两种CPU架构的镜像，为exampleservice-1.0.0-x86.tar和exampleservice-1.0.0-aarch64.tar，其values.yaml中镜像配置：

```
image: exampleservice:1.0.0
kafka:
  image:
    registry: docker.io
    repository: bitnami/kafka
    tag: 2.5.0
zookeeper:
  image:
    repository: docker.io/bitnami/zookeeper
    tag: 3.4.0
mariadb:
  image:
    registry: docker.io
    repository: bitnami/mariadb
    tag: 10.5.8-debian-10-r46
redis:
  image: docker.io/bitnami/redis:6.0.10
```

mapping.yaml配置为：

```
instance:
  - image: kafka
    tag: 2.5.0
    address:
      endpoint: kafka.image.registry
      repo: kafka.image.repository
      tag: kafka.image.tag
    package:
```

```
    x86_64: kafka-2.5.0.tar
- image: mariadb
  tag: 10.5.8-debian-10-r46
  address:
    endpoint: mariadb.image.registry
    repo: mariadb.image.repository
    tag: mariadb.image.tag
  package:
    x86_64: mariadb-10.5.8.tar
- image: zookeeper
  tag: 3.4.0
  address:
    repository: zookeeper.image.repository
    tag: zookeeper.image.tag
  package:
    x86_64: zookeeper-3.4.14.tar
- image: redis
  tag: 6.0.10
  address:
    fullAddress: redis.image
  package:
    x86_64: exampleservice-1.0.0-x86.tar
    aarch64: exampleservice-1.0.0-arm64.tar
```

Operator 服务示例

Operator服务镜像配置在*.clusterserviceversion.yaml文件中，分布在两处：

1. 作为Operator运维功能的实例镜像，其默认配置在spec.install中。
2. 作为真正提供服务功能的实例镜像，其默认配置在metadata.annotations.alm-examples中。

开发者在mapping.yaml中指定需要在*.clusterserviceversion.yaml中替换的镜像地址，以etcd operator为例：

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
metadata:
annotations:
alm-examples: "[
  {
    "apiVersion": "etcd.database.coreos.com/v1beta2",
    "kind": "EtcdCluster",
    "metadata": {
      "name": "example"
    },
    "spec": {
      "size": 3,
      "version": "3.2.13",
      "cluster": {
        "registry": "quay.io/coreos/ “,
        "repo": "etcd-operator@sha256",
        "tag": 66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b"
      },
    }
  },
  {
    "apiVersion": "etcd.database.coreos.com/v1beta2",
    "kind": "EtcdRestore",
    "metadata": {
      "name": "example-etcd-cluster-restore"
    },
    "spec": {
      "etcdCluster": {
        "name": "example-etcd-cluster"
      },
      "backupStorageType": "S3",
```

```
      "restore": {
        "repo": "quay.io/coreos/etcd-operator@sha256",
        "tag": 66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b"
      },
      "s3": {
        "path": "<full-s3-path>",
        "awsSecret": "<aws-secret>"'
      }
    }
  },
  {
    "apiVersion": "etcd.database.coreos.com/v1beta2",
    "kind": "EtcdBackup",
    "metadata": {
      "name": "example-etcd-cluster-backup"
    },
    "spec": {
      "etcdEndpoints": ["<etcd-cluster-endpoints>"],
      "storageType":"S3",
      "image": "quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b",
      "s3": {
        "path": "<full-s3-path>",
        "awsSecret": "<aws-secret>"
      }
    }
  }
]"
capabilities: Full Lifecycle
categories: Database
containerImage: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
createdAt: 2019-02-28 01:03:00
description: Create and maintain highly-available etcd clusters on Kubernetes
repository: https://github.com/coreos/etcd-operator
tectonic-visibility: ocs
name: etcdoperator.v0.9.4
namespace: placeholder
spec:
  install:
    spec:
      deployments:
      - name: etcd-operator
        spec:
          replicas: 1
          selector:
            matchLabels:
              name: etcd-operator-alm-owned
          template:
            metadata:
              labels:
                name: etcd-operator-alm-owned
              name: etcd-operator-alm-owned
            spec:
              containers:
              - command:
                - etcd-operator
                - --create-crd=false
                env:
                - name: MY_POD_NAMESPACE
                  valueFrom:
                    fieldRef:
                      fieldPath: metadata.namespace
                - name: MY_POD_NAME
                  valueFrom:
                    fieldRef:
                      fieldPath: metadata.name
                image: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
```

```
            name: etcd-operator
          - command:
            - etcd-backup-operator
            - --create-crd=false
            env:
            - name: MY_POD_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: MY_POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            image: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
            name: etcd-backup-operator
          - command:
            - etcd-restore-operator
            - --create-crd=false
            env:
            - name: MY_POD_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: MY_POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            image: quay.io/coreos/etcd-
operator@sha256:66a37fd61a06a43969854ee6d3e21087a98b93838e284a6086b13917f96b0d9b
            name: etcd-restore-operator
          serviceAccountName: etcd-operator
```

etcd operator中包含多个镜像，etcd-operator-x86-0.9.4.tar, etcd-operator-arm64-0.9.4.tar, etcd-backup-operator-x86-0.9.4.tar, etcd-backup-operator-arm64-0.9.4.tar, etcd-restore-operator-x86-0.9.4.tar, etcd-restore-operator-arm64-0.9.4.tar，除了instances字段，operator服务的mapping.yaml还多了一个operator字段。

```
lifecycle:
  - deployment: etcd-operator
    containers:
      - type: container
        name: etcd-operator
        image: etcd-operator
        tag: 0.9.4
        package:
          x86_64: etcd-operator-0.9.4.tar
          aarch64: etcd-operator-0.9.4.tar
      - type: container
        name: etcd-backup-operator
        image: etcd-backup-operator
        tag: 0.9.4
        package:
          x86_64: etcd-backup-operator-x86-0.9.4.tar
          aarch64: etcd-backup-operator-arm64-0.9.4.tar
      - type: container
        name: etcd-restore-operator
        image: etcd-restore-operator
        tag: 0.9.4
        package:
          x86_64: etcd-restore-operator-x86-0.9.4.tar
          aarch64: etcd-restore-operator-arm64-0.9.4.tar
instance:
  - kind: EtcdCluster
    image: etcd
    tag: 0.9.4
    address:
      endpoint: spec.cluster.registry
```

```
    repo: spec.cluster.repo
    tag: spec.cluster.tag
  package:
    x86_64: etcd-x86-0.9.4.tar
    aarch64: etcd-operator-arm64-0.9.4.tar
- kind: EtcdRestore
  image: etcd-restore
  tag: 0.9.4
  address:
    repository: spec.restore.repo
    tag: spec.restore.tag
  package:
    x86_64: etcd-x86-0.9.4.tar
    aarch64: etcd-operator-arm64-0.9.4.tar
- kind: EtcdBackup
  image: etcd-restore
  tag: 0.9.5
  address:
    fullAddress: spec.image
  package:
    x86_64: etcd-restore-x86-0.9.6.tar
    aarch64: etcd-restore-arm64-0.9.4.tar
```

mapping.yaml中instances字段支持列表形式，根据*.clusterserviceversion.yaml中的alm-example来配置，每个kind对应instances字段下一个列表元素。mapping.yaml中operator字段也支持列表的形式，对应*.clusterserviceversion.yaml中的spec.install.spec.deployments下的deployment列表。

须知

*.clusterserviceversion.yaml中spec.install.spec.deployments每个deployment的镜像配置都只能配置完整的镜像地址，不能再以fullAddress，repository，tag等字段的方式来拼接，因此无需再指定被替换的镜像地址，osc根据package字段中的内容来回填的image， type分为initContainer/container/ephemeralContainer三种类型，对应deployment中支持的三种container类型场景状态。

上述例子中lifecycle字段只是部分配置，deployment元素的完整配置如下：

```
lifecycle:
  - deployment: xxx                          # deployment的名字
    containers:
      - type: initContainer                  # deployment下的container类型，主要分为initContainer/container/
ephemeralContainer三种
        name: xxx                            # container类型的name值
        image: xxx                           # 定义推送到仓库后的镜像名称，名称必须符合docker镜像命名规范
        tag: xxx                             # 定义推送到仓库后的镜像版本tag
        package:
         x86_64: xxx.tar
          aarch64: xxx.tar
        env:                                 # container env字段下变量中涉及需要进行镜像地址替换的声明，没有可不
填
          - name: XXX                        # env中环境name变量的名称
            image: xxx                       # 定义推送到仓库后的镜像名称，名称必须符合docker镜像命名规范
            tag: xxx                         # 定义推送到仓库后的镜像版本tag
            package:
              x86_64: xxx.tar
              aarch64: xxx.tar
      - type: container
        name: xxx
        image: xxx
        tag: xxx
        package:
         x86_64: xxx.tar
          aarch64: xxx.tar
```

```
      env:                          # container env字段下变量中涉及需要进行镜像地址替换的声明，没有可不
填
        - name: XXX                     # env中环境name变量的名称
          image: xxx                   # 定义推送到仓库后的镜像名称，名称必须符合docker镜像命名规范
          tag: xxx                    # 定义推送到仓库后的镜像版本tag
          package:
            x86_64: xxx.tar
            aarch64: xxx.tar
    - type: ephemeralContainer
      name: xxx
      image: xxx
      tag: xxx
      package:
       x86_64: xxx.tar
        aarch64: xxx.tar
      env:                          # container env字段下变量中涉及需要进行镜像地址替换的声明，没有可不
填
        - name: XXX                     # env中环境name变量的名称
          image: xxx                   # 定义推送到仓库后的镜像名称，名称必须符合docker镜像命名规范
          tag: xxx                    # 定义推送到仓库后的镜像版本tag
          package:
            x86_64: xxx.tar
            aarch64: xxx.tar
```

注意

- operator.deployment.containers下每个name对应的配置，如果是单CPU架构，x86_64和aarch64二选一，多架构则都配置，每种容器启动的镜像只能有一个。
- initContainers，containers，ephemeralContainers都是可选项，这些字段是*.clusterserviceversion.yaml中用来配置容器初始化的，根据实际情况配置，如果没有可以不配置。

3 服务发布

3.1 前置说明

1. 上传商用服务的账号须具有ISV权限，具体申请方式请参见如何加入华为云云市场。
2. 使用商用服务发布的账号需要在上海一region下创建一个OBS桶，桶创建要求如下：

表 3-1 OBS 桶创建参数说明

参数	值	说明
区域	上海一	只能选上海一，不能选其它区域，OSC虽然是Global服务，但本身部署在上海一。
默认加密	关闭	OSC无法获取到OBS加密秘钥，因此不能打开桶的默认加密策略。

说明

如果已经有可用OBS桶，可跳过此步骤，表中的参数需要按要求填写，其它的可选默认值。

3. 合作伙伴上传的服务包，需要保证元数据文件中配置provider或者maintainer字段数据，从而可以将上架服务包在市场中展示服务提供商的联系方式。
4. 用户已经阅读并了解OSC服务开发者指南和OSC服务使用者指南，已确认过需要发布的服务包符合OSC支持的规范并在具体环境验证通过后，再进行相关操作处理。

3.2 创建服务

步骤1 使用ISV账号登录OSC控制台，单击左侧导航栏“服务发布”，查看右侧服务发布流程。

步骤2 单击“创建服务”按钮，在“创建服务”弹框页面填写“名称”、“描述”，并选择模板仓库和镜像仓库的后端地址。

- 名称：填写需要上传的服务包中的名称。
- 描述：用于描述此服务的相关功能描述等。
- 模板仓库：用于存储服务包中的operator、helm chart或者osc规范包的模板文件。
- 镜像仓库：用于存储服务包中的容器镜像。如果用户没有对应的仓库组织，可单击“新建组织”跳转到对应页面进行创建。可参考创建组织。

步骤3 单击“确定”按钮，创建服务完成。

----结束

3.3 新增版本

步骤1 使用ISV账号登录OSC控制台，单击左侧导航栏“服务发布”。

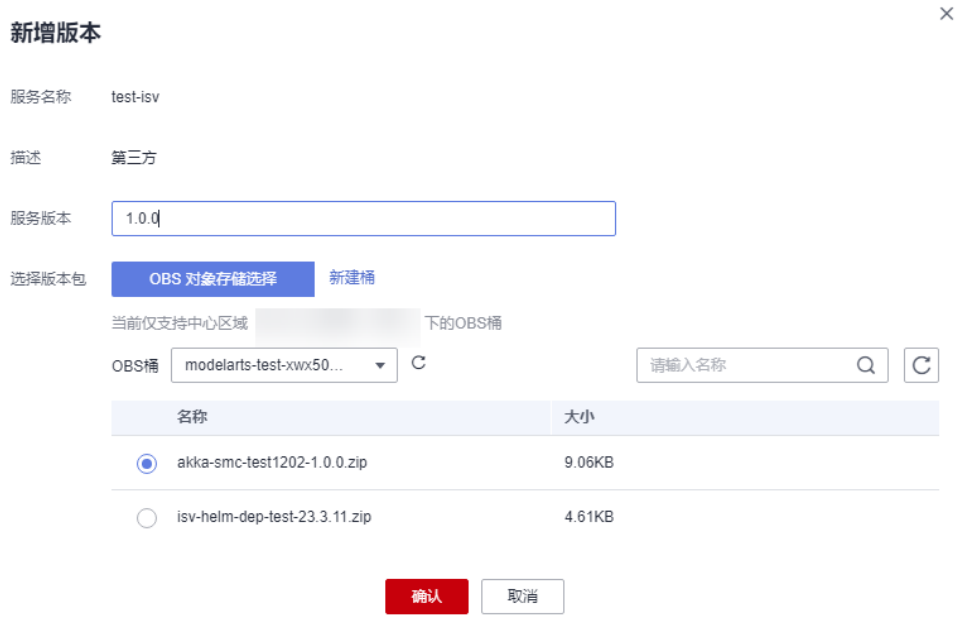
步骤2 单击新创建的服务右侧“操作“栏中的“新增版本”，进入新增版本页面。

步骤3 填写“服务版本”，选择对应的OBS桶，关联需要上传到OSC平台的服务包，完成后单击“确认”即可完成添加版本操作。

如果没有OBS桶，则可以单击“新建桶”进行跳转到对应页面进行创建并上传服务包。

说明

当前OSC仅支持中心region(上海一)下的桶对接使用。



步骤4 服务版本添加完成，服务版本转为“待发布”状态，用户可在“我的服务”-“私有服务”中查看验证。

----结束

3.4 提交验证

前置条件

用户已经在“我的服务”-“私有服务”中进行部署验证通过。

操作步骤

步骤1 使用ISV账号登录OSC控制台，单击左侧导航栏“服务发布”。

步骤2 选择需要提交验证的服务，单击下拉按钮，查看待发布的服务版本，单击右侧操作栏中的“验证”按钮，进入提交验证材料页面。

步骤3 用户可以从验证材料页面中下载模板，然后根据模板内容填写相对于的验证数据和截图。

步骤4 将填写好的材料保存后，在提交验证页面上传，等待管理员审批。

----结束

3.5 注册资产

前置条件

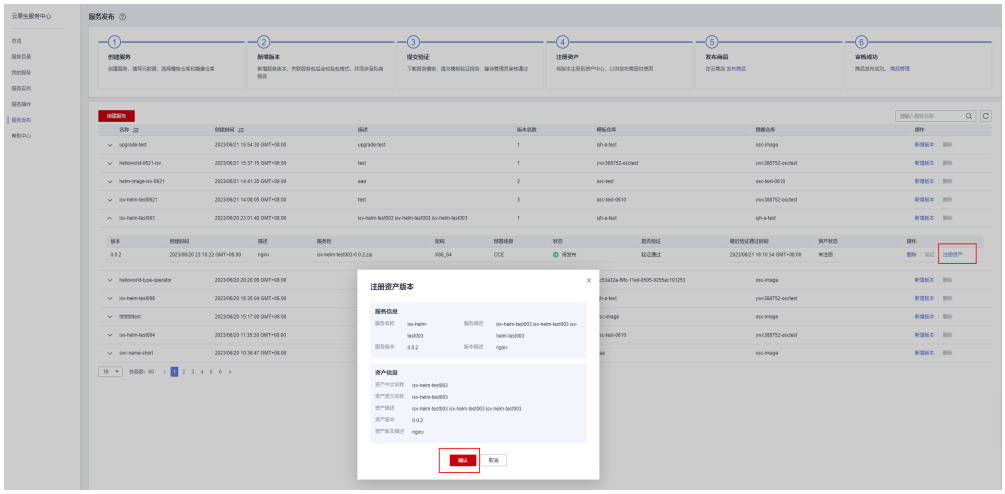
用户已经将对应服务包提交验证申请，并有管理员审批通过。

操作步骤

步骤1 使用ISV账号登录OSC控制台，单击左侧导航栏“服务发布”。

步骤2 选择需要注册资产的服务版本。

步骤3 单击对应服务版本的“注册资产”按钮，弹框查看对应的服务和默认资产信息，检查无误后，单击“确定”后进行资产注册。



步骤4 注册后的资产可以在“卖家中心”-“资产中心”查看注册的资产列表并进行管理。

----结束

3.6 发布服务

OSC负责管理服务包，商品的正式发布需要到华为云云市场**卖家中心**发布，交付方式选择”容器”，其它信息请参见**商品发布说明**。

操作步骤

步骤1 在服务发布页面单击“发布商品”链接进入云市场发布商品界面。

注意

发布服务必须使用通过云市场认证的账号登录，如果没有通过云市场认证的账号，请申请账号，具体请参见**云市场计划**。

步骤2 在“发布商品”页面商品接入类型选择“容器”，填写商品信息。

步骤3 在选择资产时，单击“增加”按钮选择OSC服务已经注册的资产类。

步骤4 单击“提交”按钮提交商品上架申请，等待云市场运营人员的审批。

步骤5 您可以在云市场“卖家中心->商品管理->我的申请”里查看商品的审批状态。

步骤6 待云市场审批完毕后，OSC服务发布页面对应的服务版本会变成“发布成功”状态，说明服务已发布成功。

----结束

3.7 上架服务新版本

服务发布上架后，将会对外提供服务。当用户的服务需要上线新特性时，则需要上架新的版本服务包，服务新版本上架流程具体步骤如下：

步骤1 使用ISV账号登录OSC控制台，单击左侧导航栏"服务发布"进入服务发布页面。

步骤2 重复执行**新增版本**到**注册资产**步骤，执行完成后，服务将会自动发布成功。

----结束