

eSDK Huawei Storage COSI Plugins V1.0.0

用户指南

文档版本	01
发布日期	2024-10-09



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

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。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://e.huawei.com>

安全声明

产品生命周期政策

华为公司对产品生命周期的规定以“产品生命周期终止政策”为准，该政策的详细内容请参见如下网址：
<https://support.huawei.com/ecolumnsweb/zh/warranty-policy>

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
<https://www.huawei.com/cn/psirt/vul-response-process>
如企业客户须获取漏洞信息，请参见如下网址：
<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

华为初始证书权责说明

华为公司对随设备出厂的初始数字证书，发布了“华为设备初始数字证书权责说明”，该说明的详细内容请参见如下网址：
<https://support.huawei.com/enterprise/zh/bulletins-service/ENEWS2000015766>

华为企业业务最终用户许可协议(EULA)

本最终用户许可协议是最终用户（个人、公司或其他任何实体）与华为公司就华为软件的使用所缔结的协议。最终用户对华为软件的使用受本协议约束，该协议的详细内容请参见如下网址：
<https://e.huawei.com/cn/about/eula>






产品资料生命周期策略

华为公司针对随产品版本发布的售后客户资料（产品资料），发布了“产品资料生命周期策略”，该策略的详细内容请参见如下网址：
<https://support.huawei.com/enterprise/zh/bulletins-website/ENEWS2000017760>

前言

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
01	2024-10-09	第一次正式发布。

目录

前言..... iii

1 概述..... 1

2 兼容性和特性..... 2

2.1 Kubernetes 兼容性..... 2

2.2 华为存储兼容性..... 2

2.3 特性矩阵..... 3

3 安装前准备..... 4

3.1 获取工具..... 4

3.2 获取华为 COSI 软件包..... 5

3.3 上传华为 COSI 镜像..... 5

3.3.1 上传镜像到镜像仓库..... 5

3.3.2 上传镜像到本地节点..... 6

3.4 检查 COSI 依赖的镜像..... 7

3.5 安装 Helm..... 7

3.6 准备配置文件..... 8

4 安装部署..... 12

4.1 安装软件..... 12

4.2 卸载软件..... 13

4.3 更新软件..... 14

4.4 更新回退..... 14

4.5 升级软件..... 15

4.6 升级回退..... 16

5 使用华为 COSI..... 17

5.1 桶管理..... 17

5.1.1 动态桶供应..... 17

5.1.1.1 配置存储业务面账户信息的 Secret..... 17

5.1.1.2 配置 BucketClass..... 18

5.1.1.3 配置 BucketClaim..... 20

5.1.2 静态桶供应..... 22

5.1.2.1 配置存储业务面账户信息的 Secret..... 22

5.1.2.2 配置 BucketClass..... 23

5.1.2.3 配置 Bucket..... 25

5.1.2.4 配置 BucketClaim..... 27

5.1.3 桶回收..... 29

5.2 桶凭证管理..... 29

5.2.1 发放桶凭证..... 29

5.2.1.1 配置存储管理面账户信息的 Secret..... 30

5.2.1.2 配置 BucketAccessClass..... 31

5.2.1.3 配置 BucketAccess..... 33

5.2.2 回收桶凭证..... 35

6 安全加固..... 37

6.1 华为 COSI 容器最小化权限运行参数配置指导..... 37

7 FAQs..... 39

7.1 如何下载容器镜像到本地..... 39

7.2 如何查看华为 COSI 日志..... 40

7.3 如何获取 COSI 版本信息..... 41

7.4 COSI Sidecar 和 Controller 组件社区问题..... 41

1 概述

容器对象存储接口（[Container Object Storage Interface](#)），简称COSI，是 Kubernetes生态下的一组用于配置和管理对象存储的抽象标准接口。它的目标是成为多个对象存储供应商的通用抽象层，这样工作负载就可以请求并自动配置对象存储桶。

2 兼容性和特性

- 2.1 Kubernetes兼容性
- 2.2 华为存储兼容性
- 2.3 特性矩阵

2.1 Kubernetes 兼容性

表 2-1 支持的容器管理平台

容器管理平台	版本
Kubernetes	1.25~1.30
Red Hat OpenShift Container Platform	4.13, 4.14, 4.15

须知

- 本文中的所有命令以Kubernetes容器管理平台为例，如果是在OpenShift平台上安装使用华为CDR服务，请使用oc命令替换kubect命令。例如：将**kubect get pods -n default**命令替换为**oc get pods -n default**命令。

2.2 华为存储兼容性

表 2-2 存储兼容性

存储产品	版本
OceanStor Pacific 系列	8.1.5, 8.2.0

2.3 特性矩阵

表 2-3 支持的特性与 Kubernetes 版本

特性	V1.25+
Static Bucket Provisioning	√
Dynamic Bucket Provisioning	√
Bucket Access Granting	√
Bucket Access Revoking	√

表 2-4 支持的特性与协议

特性	AWS S3	GCS	Azure Blob
Static Bucket Provisioning	√	x	x
Dynamic Bucket Provisioning	√	x	x
Bucket Access Granting	√	x	x
Bucket Access Revoking	√	x	x

3 安装前准备

- 3.1 获取工具
- 3.2 获取华为COSI软件包
- 3.3 上传华为COSI镜像
- 3.4 检查COSI依赖的镜像
- 3.5 安装Helm
- 3.6 准备配置文件

3.1 获取工具

软件安装、配置和调测过程中，需要准备必要的工具，如表3-1所示。

表 3-1 工具列表

工具名称	工具说明	获取方式
PuTTY	跨平台远程访问工具 用于在软件安装过程中在Windows系统上访问各节点。	您可以访问chiark主页下载PuTTY软件。 低版本的PuTTY软件可能导致登录存储系统失败，建议使用最新版本的PuTTY软件。
WinSCP	跨平台文件传输工具，请使用5.7.5或更高版本，并在传输文件时选用SCP协议。 用于在Windows系统和Linux系统间传输文件。	您可以访问WinSCP主页下载WinSCP软件。

3.2 获取华为 COSI 软件包

步骤1 在开始部署服务前，用户需要准备好用于安装的COSI软件安装包，如表3-2所示。以下以eSDK_Huawei_Storage_COSI_V1.0.0_X86_64.zip软件包为例。

表 3-2 软件包列表

软件包名称	说明	获取路径
eSDK_Huawei_Storage_COSI_V1.0.0_X86_64.zip	COSI软件安装包	https://github.com/Huawei/cosi/releases
eSDK_Huawei_Storage_COSI_V1.0.0_ARM_64.zip		

步骤2 执行unzip /opt/软件包名称命令，解压软件包。其中，软件包名称为获取到的软件包。解压后软件包组件结构如表3-3。

```
# unzip /opt/eSDK_Huawei_Storage_COSI_V1.0.0_X86_64.zip -d /opt/huawei-cosi
```

表 3-3 软件包组件描述

组件	组件描述
image/	华为COSI提供的镜像。
helm/	Helm工程，用于部署华为COSI。
examples/	华为COSI使用过程中的yaml示例文件。

----结束

3.3 上传华为 COSI 镜像

为了后续在容器管理平台中可以使用COSI镜像，需要按照以下方式中的一种提前将COSI镜像导入到集群中：

- 使用Docker工具，将COSI镜像上传至镜像仓库（推荐）。
- 手动将COSI镜像导入到所有需要部署华为COSI的节点。

3.3.1 上传镜像到镜像仓库

前提条件

已准备一台已安装Docker的Linux主机，且该主机支持访问镜像仓库。

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录已安装Docker的Linux主机。

步骤2 参考[3.2 获取华为COSI软件包](#)步骤获取软件包，并进入到image工作目录下。

步骤3 执行docker load -i huawei-cosi-driver-1.0.0.tar命令，将COSI Driver镜像导入当前节点。

```
# docker load -i huawei-cosi-driver-1.0.0.tar
Loaded image: huawei-cosi-driver:1.0.0
```

步骤4 执行docker tag huawei-cosi-driver:1.0.0 repo.huawei.com/huawei-cosi-driver:1.0.0命令，添加镜像仓库地址到镜像标签。其中repo.huawei.com表示镜像仓库的地址。

```
# docker tag huawei-cosi-driver:1.0.0 repo.huawei.com/huawei-cosi-driver:1.0.0
```

步骤5 执行docker push repo.huawei.com/huawei-cosi-driver:1.0.0命令，将COSI镜像上传到镜像仓库。其中repo.huawei.com表示镜像仓库的地址。

```
# docker push repo.huawei.com/huawei-cosi-driver:1.0.0
```

步骤6 执行docker load -i huawei-cosi-liveness-probe-1.0.0.tar命令，将COSI Driver镜像导入当前节点。

```
# docker load -i huawei-cosi-liveness-probe-1.0.0.tar
Loaded image: huawei-cosi-liveness-probe:1.0.0
```

步骤7 执行docker tag huawei-cosi-liveness-probe-1.0.0 repo.huawei.com/huawei-cosi-liveness-probe:1.0.0命令，添加镜像仓库地址到镜像标签。其中repo.huawei.com表示镜像仓库的地址。

```
# docker tag huawei-cosi-liveness-probe:1.0.0 repo.huawei.com/huawei-cosi-liveness-probe:1.0.0
```

步骤8 执行docker push repo.huawei.com/huawei-cosi-liveness-probe:1.0.0命令，将COSI镜像上传到镜像仓库。其中repo.huawei.com表示镜像仓库的地址。

```
# docker push repo.huawei.com/huawei-cosi-liveness-probe:1.0.0
```

----结束

3.3.2 上传镜像到本地节点

若镜像已上传至镜像仓库，则跳过本章节。

前提条件

- 该节点已经安装Docker或其他容器引擎。

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录需要导入镜像的节点。

步骤2 参考[3.2 获取华为COSI软件包](#)步骤获取软件包，并进入到image工作目录下。

步骤3 执行命令依次将image目录下的所有华为COSI镜像导入至本地节点，其中name参数是镜像tar包的名字。

使用Docker容器引擎执行：

```
# docker load -i <name>.tar
```

使用containerd容器引擎执行：

```
# ctr -n k8s.io image import <name>.tar
```

使用Podman容器引擎执行：

```
# podman load -i <name>.tar
```

须知

当节点主机安装的是其他容器引擎时，请使用对应容器引擎的导入镜像命令。

----结束

3.4 检查 COSI 依赖的镜像

华为COSI安装过程中需要依赖下表中的镜像，若集群中的所有worker节点已连接互联网且能够在线拉取镜像，则可跳过本章节。若集群中的节点无法连接互联网，则请根据使用的Kubernetes版本，下载对应的镜像文件并上传到镜像仓库中或者导入Kubernetes集群的worker节点中。

表 3-4 Huawei COSI 依赖的镜像

容器名称	容器镜像	功能描述
cosi-controller	gcr.io/k8s-staging-sig-storage/objectstorage-controller:latest	Kubernetes社区提供，管理BucketClaim对象的生命周期。
cosi-sidecar	gcr.io/k8s-staging-sig-storage/objectstorage-sidecar:latest	Kubernetes社区提供，管理Bucket和BucketAccess 对象的生命周期。
huawei-cosi-driver	huawei-cosi-driver:1.0.0	华为COSI软件包提供，用于提供华为COSI支持的所有特性。
livenessprobe	huawei-cosi-liveness-probe:1.0.0	华为COSI软件包提供，用于提供华为COSI驱动的健康探测功能。

说明

如何下载容器镜像到本地主机，请参考[7.1 如何下载容器镜像到本地](#)。

3.5 安装 Helm

说明

当前仅支持Helm 3。

Helm是Kubernetes生态系统中的一个软件包管理工具，类似Ubuntu的APT（Advanced Packaging Tool）、CentOS的YUM（Yellowdog Updater, Modified）、或Python的PIP（Package Installer for Python）一样，Helm专门负责管理Kubernetes的应用资源。使用Helm可以对Kubernetes应用进行统一打包、分发、安装、升级以及回退等操作。

- Helm的获取、安装：[点此前往](#)。
- Helm的其他信息请参考：[点此前往](#)。

3.6 准备配置文件

在使用Helm时，需要根据部署时对接的华为存储以及需要使用的特性准备values.yaml文件。

操作步骤

- 步骤1** 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。
- 步骤2** 执行`cd /opt/huawei-cosi/helm/`命令，进入到Helm的工作目录。
- 步骤3** 执行`vi values.yaml`命令，配置values.yaml中的配置项参数。修改完成后，按`Esc`，并输入`:wq!`，保存修改。相关参数说明如表3-5、表3-6、表3-7所示。
- global配置项主要配置系统需要的全局信息。

表 3-5 global 配置项

参数	描述	必选参数	默认值
replicaCount	COSI部署的Deployment对应Pod副本数	否	1，建议不超过2。
securityContext.runAsNonRoot	COSI容器是否以非root用户运行	否	false 须知 当参数设置为true时，runAsUser和runAsGroup参数才生效。
securityContext.runAsUser	COSI容器非root运行时的用户ID	否	1000
securityContext.runAsGroup	COSI容器非root运行时的用户组ID	否	1000
securityContext.enabled	COSI容器是否以特权容器运行	否	true
logging.module	日志记录模式。 可选值： <ul style="list-style-type: none">file：COSI容器的运行日志将会持久化到容器运行的主机节点。console：COSI容器日志将以标准输出方式记录。	是	file 须知 当参数设置为file时，fileSize和maxBackups参数才生效。
logging.level	日志级别	是	info 可选值：debug，info，warning，error

参数	描述	必选参数	默认值
logging.fileSize	日志文件大小	是	20 MB
logging.maxBackups	日志最大备份数	是	9

须知

- 根据global配置项中securityContext的默认参数，华为COSI容器默认以root用户和特权容器运行。目的是在不同的容器管理平台下，都能够正常安装部署，且能够将运行日志持久化到节点主机的/var/log/huawei-cosi目录下。
- 如果对华为COSI容器的运行时安全有要求，请参考[6.1 华为COSI容器最小化权限运行参数配置指导](#)进行指导配置。

deploy配置项主要配置COSI需要的部署信息。

表 3-6 deploy 配置项

参数	描述	必选参数	默认值
cosiController.enabled	是否部署COSI Controller组件	否	true
cosiController.namespace	部署COSI Controller组件所在的命名空间	否	huawei-cosi
cosiProvisioner.namespace	部署COSI Provisioner组件所在的命名空间	否	huawei-cosi
cosiProvisioner.driverName	部署COSI Provisioner组件对应的驱动名称	否	cosi.huawei.com

images配置项主要配置COSI需要的镜像信息。

表 3-7 image 配置项

参数	描述	必选参数	默认值
driver.cosiDriver	华为COSI Driver的镜像名称	是	huawei-cosi-driver:1.0.0.
driver.livenessProbe	华为COSI livenessProbe的镜像名称	是	huawei-cosi-liveness-probe:1.0.0

参数	描述	必选参数	默认值
controller.cosiController	COSI Controller的镜像名称	是	gcr.io/k8s-staging-sig-storage/objectstorage-controller:latest
sidecar.cosiSidecar	容器监控接口镜像。	是	gcr.io/k8s-staging-sig-storage/objectstorage-sidecar:latest
images.imagePullPolicy.huaweiCosDriverImagePullPolicy	华为COSI驱动镜像的拉取策略	是	IfNotPresent
images.imagePullPolicy.huaweiCosLivenessProbeImagePullPolicy	华为COSI驱动的健康探测镜像的拉取策略	是	IfNotPresent
images.imagePullPolicy.cosiControllerImagePullPolicy	COSI Controller镜像的拉取策略	是	IfNotPresent
images.imagePullPolicy.cosiSidecarImagePullPolicy	COSI sidecar镜像的拉取策略	是	IfNotPresent

resources配置项主要配置COSI相关容器使用的资源配置。

表 3-8 resources 配置项

参数	描述	必选参数	默认值
container.cosiDriver.requests.cpu	cosiDriver容器最小CPU资源	是	50m
container.cosiDriver.requests.memory	cosiDriver容器最小内存资源	是	128Mi
container.cosiDriver.limits.cpu	cosiDriver容器最大CPU资源	是	100m
container.cosiDriver.limits.memory	cosiDriver容器最大内存资源	是	256Mi
container.cosiLivenessProbe.requests.cpu	cosiLivenessProbe容器最小CPU资源	是	10m
container.cosiLivenessProbe.requests.memory	cosiLivenessProbe容器最小内存资源	是	128Mi

参数	描述	必选参数	默认值
container.cosiLivenessProbe.limits.cpu	cosiLivenessProbe容器最大CPU资源	是	100m
container.cosiLivenessProbe.limits.memory	cosiLivenessProbe容器最大内存资源	是	128Mi
container.cosiSidecar.requests.cpu	cosiSidecar容器最小CPU资源	是	50m
container.cosiSidecar.requests.memory	cosiSidecar容器最小内存资源	是	128Mi
container.cosiSidecar.limits.cpu	cosiSidecar容器最大CPU资源	是	100m
container.cosiSidecar.limits.memory	cosiSidecar容器最大内存资源	是	512Mi
container.cosiController.requests.cpu	cosiController容器最小CPU资源	是	50m
container.cosiController.requests.memory	cosiController容器最小内存资源	是	128Mi
container.cosiController.limits.cpu	cosiController容器最大CPU资源	是	100m
container.cosiController.limits.memory	cosiController容器最大内存资源	是	512Mi

----结束

4 安装部署

[4.1 安装软件](#)

[4.2 卸载软件](#)

[4.3 更新软件](#)

[4.4 更新回退](#)

[4.5 升级软件](#)

[4.6 升级回退](#)

4.1 安装软件

前提条件

- master节点已完成Helm 3的安装。
- 已完成values.yaml文件的配置，详情请参考[3.6 准备配置文件](#)。

安装准备

OpenShift平台请根据以下命令创建SecurityContextConstraints资源：

1. 执行**vi huawei-cosi-scc.yaml**命令，创建SecurityContextConstraints文件。

```
# vi huawei-cosi-scc.yaml
allowHostDirVolumePlugin: true
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: true

apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: huawei-cosi-scc
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
fsGroup:
```

```
type: RunAsAny
users:
- system:serviceaccount:huawei-cosi:huawei-cosi-provisioner-sa
volumes:
- hostpath
- emptyDir
- persistentVolumeClaim
- secret
- configMap
```

2. 执行 **oc create -f huawei-cosi-scc.yaml** 命令，创建 SecurityContextConstraints。

```
# oc create -f huawei-cosi-scc.yaml
securitycontextconstraints.security.openshift.io/huawei-cosi-scc created
```

操作步骤

- 步骤1** 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

- 步骤2** 执行 **cd /opt/huawei-cosi/helm** 命令，进入到Helm的工作目录。

- 步骤3** 执行 **helm install huawei-cosi ./ -n huawei-cosi --create-namespace** 命令，安装COSI服务。

```
# helm install huawei-cosi ./ -n huawei-cosi --create-namespace
NAME: huawei-cosi
LAST DEPLOYED: Thu Aug 15 10:33:54 2024
NAMESPACE: huawei-cosi
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

- 步骤4** 执行 **kubectctl get pod -n huawei-cosi** 命令，检查服务是否启动。

```
# kubectctl get pod -n huawei-cosi
NAME                                READY    STATUS    RESTARTS   AGE
cosi-controller-cffb8c678-2lgj8     1/1      Running   0           5s
huawei-cosi-provisioner-77f4655456-7v5tk 3/3      Running   0           4s
```

----结束

4.2 卸载软件

前提条件

已使用Helm 3完成COSI的部署。

操作步骤

- 步骤1** 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

- 步骤2** 执行 **helm uninstall huawei-cosi -n huawei-cosi** 命令，卸载COSI服务。

```
# helm uninstall huawei-cosi -n huawei-cosi
release "huawei-cosi" uninstalled
```

- 步骤3** 执行 **kubectctl delete ns huawei-cosi** 命令，删除命名空间。

```
# kubectctl delete ns huawei-cosi
namespace "huawei-cosi" deleted
```

须知

- 删除命名空间会将该命名空间的所有资源进行清理，请慎重执行该操作。
- 若不执行删除命名空间操作，后续还要再次安装COSI软件使用时，请执行**kubectl delete lease --all -n huawei-cosi**命令清除该空间下的所有Lease，否则新安装软件时需要等待Lease对象释放主权，会存在2~3分钟无法接收业务的问题。

----结束

4.3 更新软件

使用场景

当更新华为COSI服务部署参数时，请使用本章节进行配置。

前提条件

已使用Helm 3完成COSI的部署。

操作步骤

- 步骤1** 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。
- 步骤2** 执行**cd /opt/huawei-cosi/helm**命令，进入到Helm的工作目录。
- 步骤3** 执行**helm get values huawei-cosi -n huawei-cosi -a > update-value.yaml**命令，获取原有服务配置文件。
- 步骤4** 执行**vi update-value.yaml**命令打开文件，根据更新需要更新参数值。修改完成后，按**Esc**，并输入**:wq!**，保存修改。配置详情请参考[3.6 准备配置文件](#)。
- 步骤5** 执行**helm upgrade huawei-cosi ./ -n huawei-cosi -f update-value.yaml --wait --timeout 2m**命令，更新COSI服务。回显中有Release "huawei-cosi" has been upgraded，则表示更新COSI服务成功。

```
# helm upgrade huawei-cosi ./ -n huawei-cosi -f update-value.yaml --wait --timeout 2m
Release "huawei-cosi" has been upgraded. Happy Helming!
NAME: huawei-cosi
LAST DEPLOYED: Fri Aug 30 17:07:33 2024
NAMESPACE: huawei-cosi
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

----结束

4.4 更新回退

使用场景

当想要回退华为COSI服务到更新前版本时，请使用本章节进行配置。

前提条件

- 已使用Helm 3完成COSI的部署。
- 已使用Helm 3完成华为COSI更新。

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

步骤2 执行**helm history huawei-cosi -n huawei-cosi**命令，查看Helm部署华为COSI服务的历史版本。

```
# helm history huawei-cosi -n huawei-cosi
REVISION  UPDATED              STATUS   CHART       APP VERSION  DESCRIPTION
1         Fri Aug 30 11:41:19 2024  superseded  cosi-1.0.0   1.0.0        Install complete
2         Fri Aug 30 17:07:33 2024  deployed   cosi-1.0.0   1.0.0        Upgrade complete
```

步骤3 执行**helm rollback huawei-cosi revision-number -n huawei-cosi --wait --timeout 2m**命令，回退华为COSI服务到指定版本。回显中有Rollback was a success，则表示回退华为COSI服务到指定版本成功。

其中，revision-number为**步骤2**查询到的版本号。例如版本为：1。

```
# helm rollback huawei-cosi 1 -n huawei-cosi --wait --timeout 2m
Rollback was a success! Happy Helming!
```

----结束

4.5 升级软件

使用场景

当升级华为COSI服务版本时，请使用本章节进行配置。

前提条件

已使用Helm 3完成COSI的部署。

注意事项

升级时如果values.yaml文件和update-value.yaml文件含有相同参数的配置，优先使用update-value.yaml内参数。

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

步骤2 将新的镜像导入相应节点，详情请参考[3.3 上传华为COSI镜像](#)。

步骤3 执行**cd /opt/huawei-cosi/helm**命令，进入到新的安装包下Helm的工作目录。

步骤4 执行**helm get values huawei-cosi -n huawei-cosi -a > update-value.yaml**命令，获取原有服务配置文件。

步骤5 执行**vi update-value.yaml**命令打开文件，更新镜像至指定的新版本。修改完成后，按Esc，并输入:wq!，保存修改。配置详情请参考[表3-7](#)。

步骤6 执行`helm upgrade huawei-cosi ./ -n huawei-cosi -f ./values.yaml -f update-value.yaml --wait --timeout 2m`命令，升级COSI服务。回显中有Release "huawei-cosi" has been upgraded，则表示升级COSI服务成功。

```
# helm upgrade huawei-cosi ./ -n huawei-cosi -f ./values.yaml -f update-value.yaml --wait --timeout 2m
Release "huawei-cosi" has been upgraded. Happy Helming!
NAME: huawei-cosi
LAST DEPLOYED: Fri Aug 30 17:22:30 2024
NAMESPACE: huawei-cosi
STATUS: deployed
REVISION: 4
TEST SUITE: None
```

----结束

4.6 升级回退

前提条件

- 已使用Helm 3完成COSI的部署。
- 已使用Helm 3完成COSI的升级。

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

步骤2 执行`helm history huawei-cosi -n huawei-cosi`命令，查看Helm部署华为COSI服务的历史版本。

```
# helm history huawei-cosi -n huawei-cosi
REVISION  UPDATED              STATUS      CHART          APP VERSION  DESCRIPTION
1         Fri Aug 30 11:41:19 2024  superseded  cosi-1.0.0    1.0.0        Install complete
2         Fri Aug 30 17:07:33 2024  deployed   cosi-1.0.0    1.0.0        Upgrade complete
```

步骤3 执行`helm rollback huawei-cosi revision-number -n huawei-cosi --wait --timeout 2m`命令，回退华为COSI服务到指定版本。回显中有Rollback was a success，则表示回退华为COSI服务到指定版本成功。

其中，revision-number为[步骤2](#)查询到的版本号。例如版本为：1。

```
# helm rollback huawei-cosi 1 -n huawei-cosi --wait --timeout 2m
Rollback was a success! Happy Helming!
```

----结束

5 使用华为 COSI

5.1 桶管理

5.2 桶凭证管理

5.1 桶管理

5.1.1 动态桶供应

为了完成动态桶供应，需要完成如下三步：

- 配置存储业务面账户信息的Secret
- 配置BucketClass
- 配置BucketClaim

5.1.1.1 配置存储业务面账户信息的 Secret

参考示例配置文件/opt/huawei-cosi/examples/accountsecret-service.yaml，示例如下：

```
kind: Secret
apiVersion: v1
metadata:
  name: sample-account-service-secret
  namespace: huawei-cosi
stringData:
  accessKey: <ak-value>
  secretKey: <sk-value>
  endpoint: <point-value>
```

表 5-1 Secret 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	Secret对象的名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
metadata.namespace	Secret对象的命名空间	是	-	名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
stringData.accessKey	存储侧对应账户的AK	是	-	-
stringData.secretKey	存储侧对应账户的SK	是	-	-
stringData.endpoint	存储侧业务面网络的Endpoint	是	-	支持按照域名或者IP+端口的方式进行配置。例如：https://xx.xx.xx.xx:5443。端口必须配置为5443。
data.rootCA	根证书信息，用于校验存储服务器端证书	否	-	填写Base64编码后的证书数据

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi accountsecret-service.yaml命令，参考表5-1填写示例配置文件。
- 步骤3 执行kubectl create -f accountsecret-service.yaml命令，基于准备好的yaml文件创建Secret。

kubectl create -f accountsecret-service.yaml
secret/sample-account-service-secret created
- 步骤4 执行kubectl get secret sample-account-service-secret -n huawei-cosi命令，查看当前已经创建的Secret信息。

kubectl get secret sample-account-service-secret -n huawei-cosi
NAME TYPE DATA AGE
sample-account-service-secret Opaque 3 10s
- 结束

5.1.1.2 配置 BucketClass

参考示例配置文件/opt/huawei-cosi/examples/bucketclass.yaml，示例如下：


```
kind: BucketClass
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bucket-class
driverName: cosi.huawei.com
deletionPolicy: Delete
parameters:
  accountSecretName: sample-account-service-secret
  accountSecretNamespace: huawei-cosi
  bucketACL: <bucket-acl>
  bucketLocation: <bucket-location>
```

表 5-2 BucketClass 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的 BucketClass 对象名称	是	-	支持数字、小写字母、中划线 (-) 和点 (.) 的组合，并且必须以字母数字开头和结尾，中划线 (-) 和点 (.) 不能相邻，不能有相邻的点 (.) 。 须知 最大长度建议不超过 27 个字符。否则后续使用该 BucketClass 在存储侧发放的桶资源，由于名称长度超过 63 个字符，可能会受到功能限制。
driverName	使用的驱动名称	是	-	该字段需要指定为安装华为 COSI 时设置的驱动名称。 取值和 values.yaml 配置文件中 driverName 一致。
deletionPolicy	存储侧桶资源回收策略。可选值： <ul style="list-style-type: none">• Delete• Retain	是	-	Delete：删除 BucketClaim 对象时会关联删除存储侧桶资源。 Retain：删除 BucketClaim 对象时保留存储侧桶资源。
parameters.accountSecretName	Secret 对象的名称	是	-	-
parameters.accountSecretNamespace	Secret 对象的命名空间	是	-	-

参数	说明	必选	参数默认值	备注
parameters.bucketACL	桶权限。可选值： <ul style="list-style-type: none">• private• public-read• public-read-write• authenticated-read	否	private	private：桶的所有者拥有完全控制的权限，其他任何人没有访问权限。 public-read：桶的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读的权限。 public-read-write：桶的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读和写的权限。 authenticated-read：桶的所有者拥有完全控制的权限，其他对象服务授权用户拥有读权限。
parameters.bucketLocation	桶存储区域	否	-	-

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi bucketclass.yaml命令，参考表5-2填写示例配置文件。
- 步骤3 执行kubectl create -f bucketclass.yaml命令，基于准备好的yaml文件创建BucketClass。

```
# kubectl create -f bucketclass.yaml
bucketclass.objectstorage.k8s.io/sample-bucket-class created
```
- 步骤4 执行kubectl get bucketclass sample-bucket-class命令，查看当前已经创建的BucketClass信息。

```
# kubectl get bucketclass sample-bucket-class
NAME          AGE
sample-bucket-class  10s
```

----结束

5.1.1.3 配置 BucketClaim

参考示例配置文件/opt/huawei-cosi/examples/bucketclaim.yaml，示例如下：

```
kind: BucketClaim
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bucket-claim
  namespace: huawei-cosi
spec:
```

```
bucketClassName: sample-bucket-class
protocols:
  - s3
```

表 5-3 BucketClaim 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的 BucketClaim 对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
metadata.namespace	自定义的 BucketClaim 对象的命名空间	是	-	自定义的 BucketClaim 对象的 Kubernetes命名空间。名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
spec.bucketClassName	BucketClass对象名称	是	-	-
spec.protocols	使用协议。可选值： <ul style="list-style-type: none">s3	是	-	-

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi bucketclaim.yaml命令，参考表5-3填写示例配置文件。
- 步骤3 执行kubectl create -f bucketclaim.yaml命令，基于准备好的yaml文件创建 BucketClaim。

kubectl create -f bucketclaim.yaml
bucketclaim.objectstorage.k8s.io/sample-bucket-claim created
- 步骤4 执行kubectl get bucketclaim sample-bucket-claim -n huawei-cosi -o yaml命令，查看当前已经创建的BucketClaim信息。若BucketClaim中status.bucketReady字段为true，说明BucketClaim创建成功。

kubectl get bucketclaim sample-bucket-claim -n huawei-cosi -o yaml
apiVersion: objectstorage.k8s.io/v1alpha1

```
kind: BucketClaim
metadata:
  creationTimestamp: "2024-09-25T07:10:37Z"
  finalizers:
    - cosi.objectstorage.k8s.io/bucketclaim-protection
  generation: 1
  name: sample-bucket-claim
  namespace: huawei-cosi
  resourceVersion: "166751963"
  uid: 53facdb1-9e9e-46eb-b59d-046b9982e78d
spec:
  bucketClassName: sample-bucket-class
  protocols:
    - s3
status:
  bucketName: sample-bucket-class53facdb1-9e9e-46eb-b59d-046b9982e78d
  bucketReady: true
```

----结束

5.1.2 静态桶供应

为了完成静态桶供应，需要完成如下四步：

- 配置存储业务面账户信息的Secret
- 配置BucketClass
- 配置Bucket
- 配置BucketClaim

5.1.2.1 配置存储业务面账户信息的 Secret

参考示例配置文件/opt/huawei-cosi/examples/accountsecret-service.yaml，示例如下：

```
kind: Secret
apiVersion: v1
metadata:
  name: sample-account-service-secret
  namespace: huawei-cosi
stringData:
  accessKey: <ak-value>
  secretKey: <sk-value>
  endpoint: <point-value>
```

表 5-4 Secret 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	Secret对象的名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。

参数	说明	必选	参数默认值	备注
metadata.namespace	Secret对象的命名空间	是	-	名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
stringData.accessKey	存储侧对应账户的AK	是	-	-
stringData.secretKey	存储侧对应账户的SK	是	-	-
stringData.endpoint	存储侧业务面网络的Endpoint	是	-	支持按照域名或者IP+端口的方式进行配置。例如：https://xx.xx.xx.xx:5443。端口必须配置为5443。
data.rootCA	根证书信息，用于校验存储服务器端证书	否	-	填写Base64编码后的证书数据

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi accountsecret-service.yaml命令，参考表5-4填写示例配置文件。
- 步骤3 执行kubectl create -f accountsecret-service.yaml命令，基于准备好的yaml文件创建Secret。

kubectl create -f accountsecret-service.yaml
secret/sample-account-service-secret created
- 步骤4 执行kubectl get secret sample-account-service-secret -n huawei-cosi命令，查看当前已经创建的Secret信息。

kubectl get secret sample-account-service-secret -n huawei-cosi
NAME TYPE DATA AGE
sample-account-service-secret Opaque 3 10s
- 结束

5.1.2.2 配置 BucketClass

参考示例配置文件/opt/huawei-cosi/examples/bucketclass.yaml，示例如下：

```
kind: BucketClass
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bucket-class
driverName: cos.huawei.com
deletionPolicy: Delete
parameters:
  accountSecretName: sample-account-service-secret
  accountSecretNamespace: huawei-cosi
  bucketACL: <bucket-acl>
  bucketLocation: <bucket-location>
```

表 5-5 BucketClass 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的 BucketClass 对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。 须知 最大长度建议不超过27个字符。否则后续使用该 BucketClass 在存储侧发放的桶资源，由于名称长度超过63个字符，可能会受到功能限制。
driverName	使用的驱动名称	是	-	该字段需要指定为安装华为COSI时设置的驱动名称。 取值和values.yaml配置文件中driverName一致。
deletionPolicy	存储侧桶资源回收策略。可选值： <ul style="list-style-type: none">• Delete• Retain	是	-	Delete：删除 BucketClaim 对象时会关联删除存储侧桶资源。 Retain：删除 BucketClaim 对象时保留存储侧桶资源。
parameters.accountSecretName	Secret 对象的名称	是	-	-
parameters.accountSecretNamespace	Secret 对象的命名空间	是	-	-

参数	说明	必选	参数默认值	备注
parameters.bucketACL	桶权限。可选值： <ul style="list-style-type: none">privatepublic-readpublic-read-writeauthenticated-read	否	private	private：桶的所有者拥有完全控制的权限，其他任何人没有访问权限。 public-read：桶的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读的权限。 public-read-write：桶的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读和写的权限。 authenticated-read：桶的所有者拥有完全控制的权限，其他对象服务授权用户拥有读权限。
parameters.bucketLocation	桶存储区域	否	-	-

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi bucketclass.yaml命令，参考表5-5填写示例配置文件。
- 步骤3 执行kubectl create -f bucketclass.yaml命令，基于准备好的yaml文件创建BucketClass。

```
# kubectl create -f bucketclass.yaml
bucketclass.objectstorage.k8s.io/sample-bucket-class created
```
- 步骤4 执行kubectl get bucketclass sample-bucket-class命令，查看当前已经创建的BucketClass信息。

```
# kubectl get bucketclass sample-bucket-class
NAME          AGE
sample-bucket-class  10s
```

----结束

5.1.2.3 配置 Bucket

参考示例配置文件/opt/huawei-cosi/examples/static-bucket.yaml，示例如下：

```
kind: Bucket
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-static-bucket
spec:
  bucketClaim: {}
```

```
driverName: cosi.huawei.com
bucketClassName: sample-bucket-class
existingBucketID: <account-service-secret-namespace>/<account-service-secret-name>/<storage-existing-bucket-name>
deletionPolicy: Retain
protocols:
- s3
```

表 5-6 Bucket 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的静态 Bucket对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
spec.bucketClaim	BucketClaim对象名称	是	-	此处必须设置值为：{}
spec.driverName	驱动名称	是	-	该字段需要指定为安装华为COSI时设置的驱动名称。 取值和values.yaml文件中driverName一致。
spec.bucketClassName	BucketClass对象名称	是	-	-
spec.existingBucketID	已存在的桶信息。由集群中业务面 Secret对象的命名空间，业务面 Secret对象的名称和存储侧已存在的桶名称三部分信息组成。	是	-	格式： <account-service-secret-namespace>/<account-service-secret-name>/<storage-existing-bucket-name> 例如： secret-ns/secret-name/exist-bucket
spec.deletionPolicy	存储侧桶资源回收策略。可选值： <ul style="list-style-type: none">DeleteRetain	是	Retain	Delete：删除Bucket对象时会关联删除存储侧桶资源。 Retain：删除Bucket对象时保留存储侧桶资源。
spec.protocols	使用协议。可选值： <ul style="list-style-type: none">s3	是	-	-

步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。

步骤2 执行**vi static-bucket.yaml**命令，参考表5-6填写示例配置文件。

步骤3 执行**kubectl create -f static-bucket.yaml**命令，基于准备好的yaml文件创建Bucket。

```
# kubectl create -f static-bucket.yaml
bucket.objectstorage.k8s.io/sample-static-bucket created
```

步骤4 执行**kubectl get bucket sample-static-bucket -o yaml**命令，查看当前已经创建的Bucket信息。若Bucket中status.bucketReady字段为true，说明Bucket创建成功。

```
# kubectl get bucket sample-static-bucket -o yaml
apiVersion: objectstorage.k8s.io/v1alpha1
kind: Bucket
metadata:
  creationTimestamp: "2024-09-25T07:34:26Z"
  finalizers:
    - cosi.objectstorage.k8s.io/bucket-protection
  generation: 2
  name: sample-static-bucket
  resourceVersion: "166754807"
  uid: ffc81c82-c8d1-4d48-946a-7191e52fda1a
spec:
  bucketClaim: {}
  bucketClassName: sample-bucket-class
  deletionPolicy: Retain
  driverName: cosi.huawei.com
  existingBucketID: huawei-cosi/sample-account-service-secret/bucket-xxx
  parameters:
    accountSecretName: sample-account-service-secret
    accountSecretNamespace: huawei-cosi
    bucketACL: private
  protocols:
    - s3
status:
  bucketID: huawei-cosi/sample-account-service-secret/bucket-xxx
  bucketReady: true
```

----结束

5.1.2.4 配置 BucketClaim

参考示例配置文件/opt/huawei-cosi/examples/static-bucketclaim.yaml，示例如下：

```
kind: BucketClaim
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-static-bucket-claim
  namespace: huawei-cosi
spec:
  bucketClassName: sample-bucket-class
  existingBucketName: sample-static-bucket
  protocols:
    - s3
```

表 5-7 BucketClaim 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的静态 BucketClaim 对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
metadata.namespace	自定义的静态 BucketClaim 对象的命名空间	是	-	自定义的 BucketClaim 对象的 Kubernetes命名空间。名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
spec.bucketClassName	BucketClass的名称	是	-	-
spec.existingBucketName	静态Bucket的名称	是	-	须知 避免创建多个 BucketClaim 对象绑定同一个静态Bucket对象。
spec.protocols	使用协议。可选值： <ul style="list-style-type: none">s3	是	-	-

- 步骤1 执行cd /opt/huawei-cosi/examples/命令，进入到示例文件目录。
- 步骤2 执行vi static-bucketclaim.yaml命令，参考表5-7填写示例配置文件。
- 步骤3 执行kubectl create -f static-bucketclaim.yaml命令，基于准备好的yaml文件创建 BucketClaim。

kubectl create -f static-bucketclaim.yaml
bucketclaim.objectstorage.k8s.io/sample-static-bucket-claim created
- 步骤4 执行kubectl get bucketclaim sample-static-bucket-claim -n huawei-cosi -o yaml命令，查看当前已经创建的BucketClaim信息。若BucketClaim中

status.bucketName为[5.1.2.3 配置Bucket](#)中创建的Bucket名称，且status.bucketReady字段为true，则说明BucketClaim创建成功。

```
# kubectl get bucketclaim sample-static-bucket-claim -n huawei-cosi -o yaml
apiVersion: objectstorage.k8s.io/v1alpha1
kind: BucketClaim
metadata:
  creationTimestamp: "2024-09-25T07:37:45Z"
  finalizers:
  - cosi.objectstorage.k8s.io/bucketclaim-protection
  generation: 1
  name: sample-static-bucket-claim
  namespace: huawei-cosi
  resourceVersion: "166755203"
  uid: 3e6dd528-074d-4194-9b20-46ddb409e757
spec:
  bucketClassName: sample-bucket-class
  existingBucketName: sample-static-bucket
  protocols:
  - s3
status:
  bucketName: sample-static-bucket
  bucketReady: true
```

----结束

5.1.3 桶回收

前提条件

已经创建好静态或动态Bucket，并创建好对应的BucketClaim。

操作步骤

步骤1 以名称为sample-bucket-claim的BucketClaim对象为例，执行**kubectl delete bucketclaim sample-bucket-claim -n huawei-cosi**命令回收该桶对象。

```
# kubectl delete bucketclaim sample-bucket-claim -n huawei-cosi
bucketclaim.objectstorage.k8s.io "sample-bucket-claim" deleted
```

----结束

须知

静态桶回收时，存在Bucket中的deletionPolicy参数与BucketClass中的配置不相等的情况。执行回收桶操作时，以Bucket中的deletionPolicy参数为准。

5.2 桶凭证管理

前提条件

桶的发放已经完成。

5.2.1 发放桶凭证

为了完成发放桶凭证，需要完成下面三步：

- 配置存储管理面账户信息的Secret
- 配置BucketAccessClass
- 配置BucketAccess

5.2.1.1 配置存储管理面账户信息的 Secret

参考示例配置文件/opt/huawei-cosi/examples/accountsecret-management.yaml，示例如下：

```
kind: Secret
apiVersion: v1
metadata:
  name: sample-account-management-secret
  namespace: huawei-cosi
stringData:
  accessKey: <ak-value>
  secretKey: <sk-value>
  endpoint: <point-value>
```

表 5-8 Secret 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	Secret对象的名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
metadata.names pace	Secret对象的命名空间	是	-	名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
stringData.access Key	存储侧对应账户的AK	是	-	-
stringData.secret Key	存储侧对应账户的SK	是	-	-
stringData.endpoi nt	存储侧管理面网络的Endpoint	是	-	支持按照域名或者IP+端口的方式进行配置。例如：https://xx.xx.xx.xx:9443。端口必须配置为9443。

参数	说明	必选	参数默认值	备注
data.rootCA	根证书信息，校验存储服务器端证书	否	-	填写Base64编码后的证书数据

- 步骤1** 执行`cd /opt/huawei-cosi/examples/`命令，进入到示例文件目录。
- 步骤2** 执行`vi accountsecret-management.yaml`命令，参考表5-8填写示例配置文件。
- 步骤3** 执行`kubectcl create -f accountsecret-management.yaml`命令，基于准备好的yaml文件创建Secret。
- ```
kubectcl create -f accountsecret-management.yaml
secret/sample-account-management-secret created
```

- 步骤4** 执行`kubectcl get secret sample-account-management-secret -n huawei-cosi`命令，查看当前已经创建的Secret信息。
- ```
# kubectcl get secret sample-account-management-secret -n huawei-cosi
NAME                                TYPE      DATA   AGE
sample-account-management-secret    Opaque    3       10s
```
- 结束

5.2.1.2 配置 BucketAccessClass

参考示例配置文件/opt/huawei-cosi/examples/bucketaccessclass.yaml，示例如下：

```
kind: BucketAccessClass
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bucket-access-class
driverName: cosi.huawei.com
authenticationType: Key
parameters:
  accountSecretName: sample-account-management-secret
  accountSecretNamespace: huawei-cosi
  bucketPolicyModel: rw
```

表 5-9 BucketAccessClass 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的 BucketAccess Class对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。

参数	说明	必选	参数默认值	备注
driverName	使用的驱动名称	是	-	该字段需要指定为安装华为COSI时设置的驱动名称。 取值和values.yaml文件中driverName一致。
authenticationType	授权类型。可选值： <ul style="list-style-type: none">Key	是	-	-
parameters.accountSecretName	Secret对象的名称	是	-	-
parameters.accountSecretNamespace	Secret对象的命名空间	是	-	-
parameters.bucketPolicyModel	桶策略设置。可选值： <ul style="list-style-type: none">rorw	否	rw	ro：读模式的桶策略，包含以下s3操作： s3:GetObject, s3:GetObjectVersion, s3:ListMultipartUploadParts, s3:GetObjectAcl, s3:GetObjectVersionAcl, s3:ListBucketVersions, s3:ListBucket, s3:ListBucketMultipartUploads rw：读写模式的桶策略，包含以下s3操作： s3:GetObject, s3:GetObjectVersion, s3:ListMultipartUploadParts, s3:GetObjectAcl, s3:GetObjectVersionAcl, s3:ListBucketVersions, s3:ListBucket, s3:ListBucketMultipartUploads, s3:AbortMultipartUpload, s3:PutObjectAcl, s3>DeleteObjectVersion, s3:PutObjectVersionAcl, s3:PutObject,s3>DeleteObject

- 步骤1** 执行`cd /opt/huawei-cosi/examples/`命令，进入到示例文件目录。
- 步骤2** 执行`vi bucketaccessclass.yaml`命令，参考表5-9填写示例配置文件。
- 步骤3** 执行`kubectl create -f bucketaccessclass.yaml`命令，基于准备好的yaml文件创建BucketAccessClass。

```
# kubectl create -f bucketaccessclass.yaml
bucketclass.objectstorage.k8s.io/sample-bucket-access-class created
```

- 步骤4** 执行`kubectl get bucketaccessclass sample-bucket-access-class`命令，查看当前已经创建的BucketAccessClass信息。

```
# kubectl get bucketaccessclass sample-bucket-access-class
NAME                                AGE
sample-bucket-access-class          10s
```

----结束

5.2.1.3 配置 BucketAccess

参考示例配置文件/opt/huawei-cosi/examples/bucketaccess.yaml，示例如下：

```
kind: BucketAccess
apiVersion: objectstorage.k8s.io/v1alpha1
metadata:
  name: sample-bucket-access
  namespace: huawei-cosi
spec:
  bucketClaimName: sample-bucket-claim
  bucketAccessClassName: sample-bucket-access-class
  credentialsSecretName: sample-cred-secret
  protocol: s3
```

表 5-10 BucketAccess 配置参数说明

参数	说明	必选	参数默认值	备注
metadata.name	自定义的BucketAccess对象名称	是	-	支持数字、小写字母、中划线（-）和点（.）的组合，并且必须以字母数字开头和结尾，中划线（-）和点（.）不能相邻，不能有相邻的点（.）。最大长度不超过63个字符。
metadata.namespace	BucketAccess对象所在的命名空间	是	-	名称必须由小写字母、数字和“-”组成，例如：my-name、123-abc。
spec.bucketClaimName	需要发放凭证的BucketClaim对象名称	是	-	-
spec.bucketAccessClassName	需要使用的BucketAccessClass对象名称	是	-	-

参数	说明	必选	参数默认值	备注
spec.credentialSecretName	存储发放的访问凭证信息的Secret对象名称	是	-	支持数字、小写字母、中划线(-)和点(.)的组合，并且必须以字母数字开头和结尾，中划线(-)和点(.)不能相邻，不能有相邻的点(.)。最大长度不超过63个字符。 须知 <ul style="list-style-type: none">请填写在BucketAccess对象命名空间下不存在的Secret对象名称。若配置的Secret对象已经存在，则会复用该Secret对象。
spec.protocol	使用协议。可选值： <ul style="list-style-type: none">s3	是	-	-

步骤1 执行`cd /opt/huawei-cosi/examples/`命令，进入到示例文件目录。

步骤2 执行`vi bucketaccess.yaml`命令，参考表5-10填写示例配置文件。

步骤3 执行`kubectctl create -f bucketaccess.yaml`命令，基于准备好的yaml文件创建BucketAccess。

```
# kubectctl create -f bucketaccess.yaml
bucketclass.objectstorage.k8s.io/sample-bucket-access created
```

步骤4 执行`kubectctl get bucketaccess sample-bucket-access -n huawei-cosi -o yaml`命令，查看当前已经创建的BucketAccess信息。若BucketAccess中status.accessGranted字段为true，则说明BucketAccess创建成功。

```
# kubectctl get bucketaccess sample-bucket-access -n huawei-cosi -o yaml
apiVersion: objectstorage.k8s.io/v1alpha1
kind: BucketAccess
metadata:
  creationTimestamp: "2024-09-25T07:11:01Z"
  finalizers:
    - cosi.objectstorage.k8s.io/bucketaccess-protection
  generation: 1
  name: sample-bucket-access
  namespace: huawei-cosi
  resourceVersion: "166752017"
  uid: 64dd7898-5db3-4969-afce-0aee0c2cdfce
spec:
  bucketAccessClassName: sample-bucket-access-class
  bucketClaimName: sample-bucket-claim
  credentialsSecretName: sample-cred-secret
  protocol: s3
status:
  accessGranted: true
  accountID: huawei-cosi/sample-account-management-secret/ba-64dd7898-5db3-4969-afce-0aee0c2cdfce
```

步骤5 执行`kubectctl get secret sample-cred-secret -n huawei-cosi -o yaml`命令，查看生成的Secret对象详细信息。其中此次对于名称为sample-bucket-claim的

BucketClaim对象，发放的桶访问凭证信息以Base64编码的格式存储在data.BucketInfo字段中。

```
# kubectl get secret sample-cred-secret -n huawei-cosi -o yaml
apiVersion: v1
data:
  BucketInfo:
    eyJtZXRhZGF0YSI6eyJuYW1lIjoieMmtZGJjZWJlN2ltMDMzMzMyMDMTYwLThkMTYtMGMyNzcyZmQyMTk5liwiY3JlYXRpb25UaW1lc3RhbnRlc3RhcXkiOm51bGx9LCJzcGVjIjpb7ImJ1Y2tldE5hbWUiOiJzYW1wbGUtYnVja2V0LWNsYXNzMDg4OGNiOWYtYzMyYi00YjRiLWUwYmItYjA1MzNlNDg0ZjQyYliwiYXV0aGVudGljYXRpb25UeXBlljoieS2V5liwic2Vjc mV0UzM0OnsiZW5kcG9pbnQiOiJodHRwczovL3gueHgueHh4Ln4eDo1NDQzliwicmVnaW9uIjoieS2V5liwiYWNjZXNz S2V5SUQ0IiixMjM0NTY3OTg5liwiYWNjZXNzU2VjcmV0S2V5liwiMTIzNDU2Nzk4OSJ9LCJzZWNyZXRBenVyZSI 6bnVsbCwicHJvdG9jb2xzIjpbInMzll19fQ=="
kind: Secret
metadata:
  creationTimestamp: "2024-09-25T06:54:44Z"
  finalizers:
    - cosi.objectstorage.k8s.io/secret-protection
  name: sample-cred-secret
  namespace: huawei-cosi
  resourceVersion: "165711865"
  uid: 7d384522-aba9-4e87-b2c6-24f88d820fcd
type: Opaque
```

步骤6 执行**echo "<bucketInfo>" | base64 -d**命令，将Base64编码的BucketInfo信息解编码。

```
# echo
"eyJtZXRhZGF0YSI6eyJuYW1lIjoieMmtZGJjZWJlN2ltMDMzMzMyMDMTYwLThkMTYtMGMyNzcyZmQyMTk5liwiY3JlYXRpb25UaW1lc3RhbnRlc3RhcXkiOm51bGx9LCJzcGVjIjpb7ImJ1Y2tldE5hbWUiOiJzYW1wbGUtYnVja2V0LWNsYXNzMDg4OGNiOWYtYzMyYi00YjRiLWUwYmItYjA1MzNlNDg0ZjQyYliwiYXV0aGVudGljYXRpb25UeXBlljoieS2V5liwic2Vjc mV0UzM0OnsiZW5kcG9pbnQiOiJodHRwczovL3gueHgueHh4Ln4eDo1NDQzliwicmVnaW9uIjoieS2V5liwiYWNjZXNz S2V5SUQ0IiixMjM0NTY3OTg5liwiYWNjZXNzU2VjcmV0S2V5liwiMTIzNDU2Nzk4OSJ9LCJzZWNyZXRBenVyZSI 6bnVsbCwicHJvdG9jb2xzIjpbInMzll19fQ==" | base64 -d

{"metadata":{"name":"bc-dbebe7b-0333-4160-8d16-0c2772fd2199","creationTimestamp":null},"spec":
{"bucketName":"sample-bucket-class0888cb9f-c32b-4b4b-a0bb-b0533e484f42","authenticationType":"Key","secretS3":{"endpoint":"https://
x.xx.xxx.xxx:5443","region":"","accessKeyID":"1234567989","accessSecretKey":"1234567989"},"secretAzure":n
ull,"protocols":["s3"]}}
```

须知

此步骤中编码信息为模拟数据，真实数据存在敏感信息，请用户谨慎操作，避免造成数据安全问题。

----结束

5.2.2 回收桶凭证

前提条件

已经正常发放桶凭证。

操作步骤

步骤1 以名称为sample-bucket-access的BucketAccess对象为例，执行**kubectl delete bucketaccess sample-bucket-access -n huawei-cosi**命令回收该桶的访问凭证。

```
# kubectl delete bucketaccess sample-bucket-access -n huawei-cosi  
bucketaccess.objectstorage.k8s.io "sample-bucket-access" deleted
```

----**结束**

6 安全加固

6.1 华为COSI容器最小化权限运行参数配置指导

6.1 华为 COSI 容器最小化权限运行参数配置指导

背景说明

根据values.yaml文件中的global配置项securityContext的默认参数，华为COSI容器默认以root用户和特权容器运行。如果对华为COSI容器的运行时安全有要求，可参考本章节配置华为COSI容器以最小化权限运行。

以下将分两种情况说明：

- 情况1：COSI容器运行节点主机未提前规划/var/log/huawei-cosi日志目录，华为COSI容器启动时创建/var/log/huawei-cosi日志目录。
- 情况2：COSI容器运行节点主机提前规划好/var/log/huawei-cosi日志目录，华为COSI容器启动时使用/var/log/huawei-cosi日志目录。

情况 1 操作步骤

步骤1 参考表3-5和表6-1，配置华为COSI容器的运行时权限与日志记录模式。

表 6-1 容器运行时权限与支持的日志记录模式对应表

容器管理平台	容器运行时是否为root用户	是否启用特权容器	支持的日志记录模式
Kubernetes	√	√	file, console
Kubernetes	√	×	file, console
Kubernetes	×	√	console
Kubernetes	×	×	console
Red Hat OpenShift Container Platform	√	√	file, console

容器管理平台	容器运行时是否为root用户	是否启用特权容器	支持的日志记录模式
Red Hat OpenShift Container Platform	√	×	console
Red Hat OpenShift Container Platform	×	√	console
Red Hat OpenShift Container Platform	×	×	console

须知

若配置的容器运行权限参数与支持的日志记录模式参数不匹配，会出现由于权限不足导致容器无法拉起的问题。

----结束

情况 2 操作步骤

在这种情况下，华为COSI容器运行时能够以最小化权限（非root用户/非特权容器）的方式运行，且可以同时支持file和console日志记录模式。

- 步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的节点。
- 步骤2 如果容器平台Kubernetes，执行**mkdir -p /var/log/huawei-cosi && chmod 757 /var/log/huawei-cosi**创建日志目录，并且设置该日志目录的DAC权限为757。

mkdir -p /var/log/huawei-cosi && chmod 757 /var/log/huawei-cosi

如果容器平台OpenShift，执行**mkdir -p /var/log/huawei-cosi && chmod 757 /var/log/huawei-cosi && chcon -t svirt_sandbox_file_t /var/log/huawei-cosi**创建日志目录，并且设置该日志目录的DAC权限为757，SELinux权限为svirt_sandbox_file_t。

mkdir -p /var/log/huawei-cosi && chmod 757 /var/log/huawei-cosi && chcon -t svirt_sandbox_file_t /var/log/huawei-cosi
- 步骤3 重复以上步骤，在华为COSI容器运行节点提前规划/var/log/huawei-cosi日志目录。

须知

请确保华为为COSI容器可能调度的节点都已经规划好了/var/log/huawei-cosi日志目录。若当华为为COSI容器运行过程中发生了节点漂移，且新的容器运行节点未提前规划日志目录，会出现由于权限不足导致容器无法拉起的问题。

----结束

7 FAQs

- [7.1 如何下载容器镜像到本地](#)
- [7.2 如何查看华为COSI日志](#)
- [7.3 如何获取COSI版本信息](#)
- [7.4 COSI Sidecar和Controller组件社区问题](#)

7.1 如何下载容器镜像到本地

以下示例命令以k8s.gcr.io/sig-storage/livenessprobe:v2.5.0镜像为例。

使用 containerd 下载容器镜像

步骤1 执行**ctr image pull *image:tag***命令，下载镜像到本地。其中*image:tag*表示需要拉取的镜像及其标签。

```
# ctr image pull k8s.gcr.io/sig-storage/livenessprobe:v2.5.0
```

步骤2 执行**ctr image export *image.tar image:tag***命令，导出镜像到文件。其中*image:tag*表示需要导出的镜像，*image.tar*表示镜像导出后的文件名称。

```
# ctr image export livenessprobe.tar k8s.gcr.io/sig-storage/livenessprobe:v2.5.0
```

----结束

使用 Docker 下载容器镜像

步骤1 执行**docker pull *image:tag***命令，下载镜像到本地。其中*image:tag*表示需要拉去的镜像。

```
# docker pull k8s.gcr.io/sig-storage/livenessprobe:v2.5.0
```

步骤2 执行**docker save *image:tag -o image.tar***命令，导出镜像到文件。其中*image:tag*表示需要导出的镜像，*image.tar*表示镜像导出后的文件名称。

```
# docker save k8s.gcr.io/sig-storage/livenessprobe:v2.5.0 -o livenessprobe.tar
```

----结束

使用 Podman 下载容器镜像

步骤1 执行**podman pull image:tag**命令，下载镜像到本地。其中**image:tag**表示需要拉去的镜像。

```
# podman pull k8s.gcr.io/sig-storage/livenessprobe:v2.5.0
```

步骤2 执行**podman save image:tag -o image.tar**命令，导出镜像到文件。其中**image:tag**表示需要导出的镜像，**image.tar**表示镜像导出后的文件名称。

```
# podman save k8s.gcr.io/sig-storage/livenessprobe:v2.5.0 -o livenessprobe.tar
```

----结束

7.2 如何查看华为 COSI 日志

查看 huawei-cosi-provisioner 服务的持久化日志

步骤1 执行**kubectl get pods -n namespace -o wide**命令，其中**namespace**是huawei-cosi-provisioner服务部署的命名空间。根据回显找到huawei-cosi-provisioner服务部署的节点信息。

```
# kubectl get pods -n huawei-cosi -o wide
NAME                ...      NODE
huawei-cosi-provisioner-66f5747d8c-f8kxv  ...      <node-name>
```

步骤2 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群中huawei-cosi-provisioner服务所在节点。

步骤3 执行**cd /var/log/huawei-cosi/cosi-driver/**命令，进入日志目录。

```
# cd /var/log/huawei-cosi/cosi-driver/
```

步骤4 执行**vi cosi-driver**命令，查看cosi-driver容器的持久化日志。

```
# vi cosi-driver
```

步骤5 执行**vi liveness-probe**命令，查看liveness-probe容器的持久化日志。

```
# vi liveness-probe
```

----结束

查看 huawei-cosi-provisioner 服务的容器标准输出日志

步骤1 执行**kubectl get pods -n namespace -o wide**命令，其中**namespace**是huawei-cosi-provisioner服务部署的命名空间。根据回显找到huawei-cosi-provisioner服务部署的节点信息。

```
# kubectl get pods -n huawei-cosi -o wide
NAME                ...      NODE
huawei-cosi-provisioner-66f5747d8c-f8kxv  ...      <node-name>
```

步骤2 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群中huawei-cosi-provisioner服务所在节点。

步骤3 执行**cd /var/log/containers**命令，进入容器日志目录。

```
# cd /var/log/containers
```

步骤4 执行**vi huawei-cosi-provisioner-<name>_huawei-cosi_huawei-cosi-driver-<container-id>.log**命令，查看huawei-cosi-driver容器标准输出日志。

```
# vi huawei-cosi-provisioner-<name>_huawei-cosi_huawei-cosi-driver-<container-id>.log
```

须知

cosi-controller、cosi-sidecar和liveness-probe容器可参考同样的方式查看标准输出日志。

----结束

7.3 如何获取 COSI 版本信息

操作步骤

步骤1 使用远程访问工具（以PuTTY为例），通过管理IP地址，登录Kubernetes集群的任意master节点。

步骤2 执行**kubectl get cm huawei-cosi-version -n namespace -o yaml**命令，其中**namespace**是华为COSI Provisioner服务部署的命名空间。

```
# kubectl get cm huawei-cosi-version -n huawei-cosi -o yaml
apiVersion: v1
data:
  cosi-driver: 1.0.0
  liveness-probe: 1.0.0
kind: ConfigMap
metadata:
  creationTimestamp: "2024-08-16T08:18:30Z"
  name: huawei-cosi-version
  namespace: huawei-cosi
  resourceVersion: "159241105"
  uid: 689feb62-e327-4651-8db3-34417a219271
```

----结束

7.4 COSI Sidecar 和 Controller 组件社区问题

背景说明

当前COSI开源社区提供的Sidecar和Controller组件均处于alpha阶段，可能还存在较多使用问题，使用过程中遇到的问题可参考社区Issues。

社区 Issues 链接

<https://github.com/kubernetes-sigs/container-object-storage-interface-api/issues>

<https://github.com/kubernetes-sigs/container-object-storage-interface-spec/issues>

<https://github.com/kubernetes-sigs/container-object-storage-interface-controller/issues>

<https://github.com/kubernetes-sigs/container-object-storage-interface-provisioner-sidecar/issues>