

容器镜像服务

用户指南

文档版本 01
发布日期 2022-03-24



版权所有 © 华为技术有限公司 2022。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。

目录

| | |
|----------------------|-----------|
| 1 产品介绍 | 1 |
| 1.1 什么是容器镜像服务 | 1 |
| 1.2 产品优势 | 2 |
| 1.3 应用场景 | 2 |
| 1.4 基本概念 | 3 |
| 1.5 约束与限制 | 4 |
| 1.6 与其他云服务的关系 | 4 |
| 2 欢迎使用容器镜像服务 | 6 |
| 3 容器引擎基础知识 | 7 |
| 4 镜像管理 | 10 |
| 4.1 客户端上传镜像 | 10 |
| 4.2 获取长期有效登录指令 | 11 |
| 4.3 页面上传镜像 | 12 |
| 4.4 下载镜像 | 13 |
| 4.5 编辑镜像属性 | 14 |
| 4.6 共享私有镜像 | 16 |
| 4.7 添加触发器 | 18 |
| 5 组织管理 | 21 |
| 6 授权管理 | 24 |
| 7 审计 | 28 |
| 7.1 支持云审计的关键操作 | 28 |
| 7.2 查看云审计日志 | 29 |
| 8 最佳实践 | 31 |
| 8.1 编写高效的 Dockerfile | 31 |
| 9 常见问题 | 39 |
| 9.1 通用类 | 39 |
| 9.1.1 产品咨询 | 39 |
| 9.1.2 如何制作容器镜像? | 39 |
| 9.1.3 如何制作镜像压缩包? | 43 |
| 9.2 镜像管理类 | 43 |

| | |
|-------------------------------------|-----------|
| 9.2.1 长期有效的登录指令与临时登录指令的区别是什么? | 44 |
| 9.2.2 为什么通过客户端和页面上传的镜像大小不一样? | 44 |
| 9.2.3 控制台页面的容器镜像可以下载到本地吗? | 44 |
| 9.3 故障类..... | 45 |
| 9.3.1 为什么登录指令执行失败? | 45 |
| 9.3.2 为什么使用客户端上传镜像失败? | 46 |
| 9.3.3 为什么通过页面上传镜像失败? | 47 |
| 9.3.4 为什么 docker pull 指令执行失败? | 48 |
| A 修订记录..... | 50 |

1 产品介绍

1.1 什么是容器镜像服务

容器镜像服务（SoftWare Repository for Container，简称SWR）是一种支持镜像全生命周期管理的服务，提供简单易用、安全可靠的镜像管理功能，帮助您快速部署容器化服务。

容器镜像服务可配合云容器引擎CCE使用，也可单独作为容器镜像仓库使用。

图 1-1 SWR 使用流程



产品功能

- **镜像全生命周期管理**
容器镜像服务支持镜像的全生命周期管理，包括镜像的上传、下载、删除等。
- **私有镜像仓库**
容器镜像服务提供私有镜像库，并支持细粒度的权限管理，可以为不同用户分配相应的访问权限（读取、编辑、管理）。
- **大规模镜像分发P2P加速**
容器镜像服务通过镜像下载加速技术，使CCE集群下载镜像时在确保高并发下能获得更快的下载速度。
- **镜像仓库触发器**
容器镜像服务支持容器镜像版本更新自动触发部署。您只需要为镜像设置一个触发器，通过触发器，可以在每次镜像版本更新时，自动更新使用该镜像部署的应用。

如何访问容器镜像服务

云平台提供了Web化的服务管理平台（即管理控制台）和基于HTTPS请求的API（Application programming interface）管理方式。

- API方式
如果用户需要将容器镜像服务集成到第三方系统，用于二次开发，请使用API方式访问容器镜像服务。具体操作请参见《容器镜像服务API参考》。
- 管理控制台方式
其他相关操作，请使用管理控制台方式访问容器镜像服务。

1.2 产品优势

简单易用

- 无需自行搭建和运维，即可快速推送拉取容器镜像。
- 容器镜像服务的管理控制台简单易用，支持镜像的全生命周期管理。

安全可靠

- 容器镜像服务遵循HTTPS协议保障镜像安全传输，提供帐号间、帐号内多种安全隔离机制，确保用户数据访问的安全。
- 容器镜像服务依托专业存储服务，确保镜像存储更可靠。

镜像加速

- 容器镜像服务通过P2P镜像下载加速技术，使CCE集群下载镜像时在确保高并发下能获得更快的下载速度。
- 容器镜像服务智能调度全球构建节点，根据所使用的镜像地址自动分配至最近的主机节点进行构建，加速镜像的获取效率。

1.3 应用场景

镜像生命周期管理

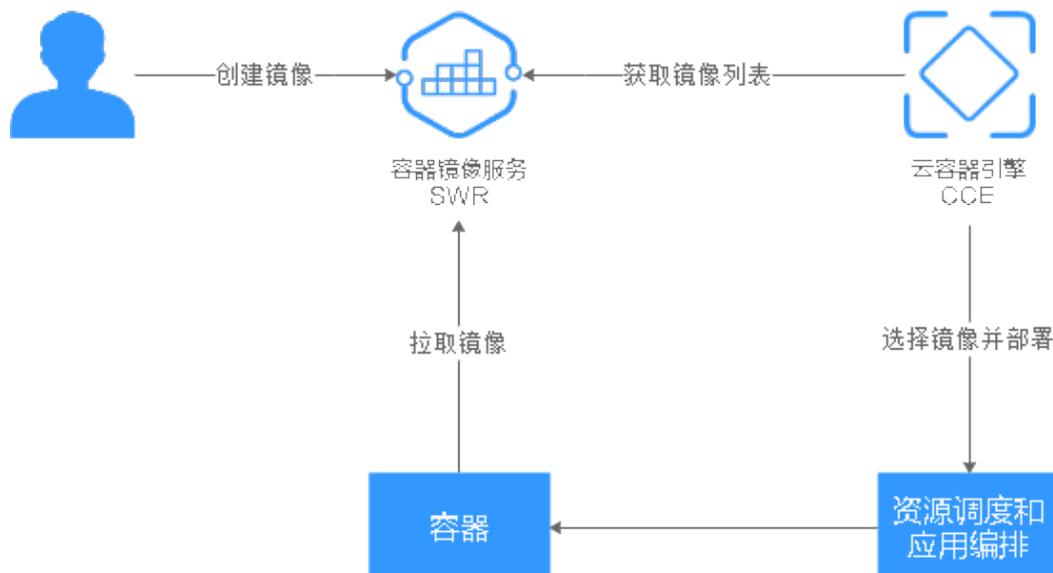
提供镜像构建、镜像上传、下载、同步、删除等完整的生命周期管理能力。

优势

- P2P加速下载：利用P2P镜像下载加速技术，提升CCE集群拉取镜像的速度。
- 高可靠的存储：依托OBS专业存储，确保镜像的存储可靠性高达11个9。
- 更安全的存储：细粒度的授权管理，让用户更精准的控制镜像访问权限。

建议搭配使用

云容器引擎CCE



1.4 基本概念

镜像 (Image)

镜像是一个模板，是容器应用打包的标准格式，在部署容器化应用时可以指定镜像，镜像可以来自于Docker镜像中心或者用户的私有仓库。例如一个镜像可以包含一个完整的Ubuntu操作系统环境，里面仅安装了用户需要的应用程序及其依赖文件。Docker镜像用于创建Docker容器。Docker本身提供了一个简单的机制来创建新的镜像或者更新已有镜像，您也可以下载其他人已经创建好的镜像来使用。

容器 (Container)

一个通过Docker镜像创建的运行实例，一个节点可运行多个容器。容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的命名空间。

镜像 (Image) 和容器 (Container) 的关系，就像是面向对象程序设计中的类和实例一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

镜像仓库 (Repository)

镜像仓库 (Repository) 用于存放Docker镜像。单个镜像仓库可对应单个具体的容器应用，并托管该应用的不同版本。

组织

组织用于隔离镜像仓库，每个组织可对应一个公司或部门，将其拥有的镜像集中在该组织下。同一用户可属于不同的组织。支持为帐号下不同用户分配相应的访问权限（读取、编辑、管理）。

图 1-2 组织



1.5 约束与限制

配额

容器镜像服务对单个用户的组织数量限定了配额。当前容器镜像服务的配额如表1-1所示。

表 1-1 容器镜像服务配额

| 资源类型 | 配额 |
|------|----|
| 组织 | 5 |

上传镜像限制

- 使用客户端上传镜像，镜像的每个layer大小不能超过10G。
- 使用页面上传镜像，每次最多上传10个文件，单个文件大小（含解压后）不得超过2G。

1.6 与其他云服务的关系

容器镜像服务需要与其他云服务协同工作，容器镜像服务和其他云服务的关系如图1-3所示。

图 1-3 容器镜像服务和其他云服务的关系



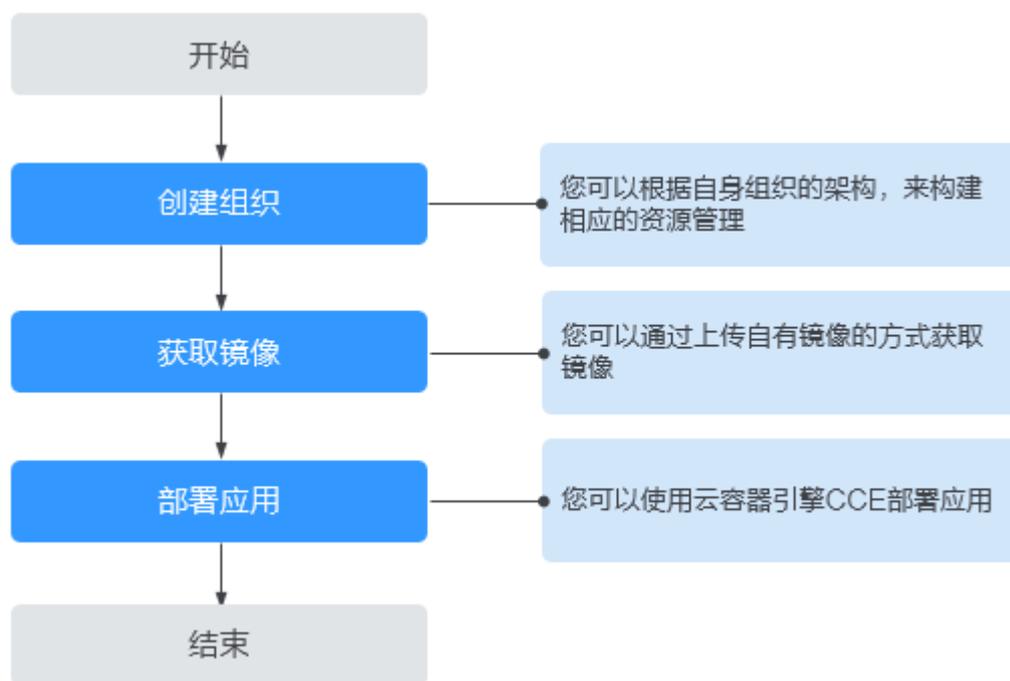
- 云容器引擎
云容器引擎（Cloud Container Engine，简称CCE）提供高可靠高性能的企业级容器应用管理服务，支持Kubernetes社区原生应用和工具，简化云上自动化容器运行环境搭建。
容器镜像服务能无缝对接CCE，您可以将容器镜像服务中的镜像部署到CCE中。
- 云审计服务
云审计服务（Cloud Trace Service，简称CTS）为您提供云服务资源的操作记录，记录内容包括您从管理控制台或者开放API发起的云服务资源操作请求以及每次请求的结果，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。
通过CTS，您可以记录与容器镜像服务相关的操作事件，便于日后的查询、审计和回溯。

2 欢迎使用容器镜像服务

容器镜像服务（SoftWare Repository for Container，简称SWR）是一种支持镜像全生命周期管理的服务，提供简单易用、安全可靠的镜像管理功能，帮助您快速部署容器化服务。

SWR提供私有镜像库，并支持细粒度的权限管理，可以为不同用户分配相应的访问权限（读取、编辑、管理）。SWR还支持容器镜像版本更新自动触发部署。您只需要为镜像设置一个触发器，通过触发器，可以在每次镜像版本更新时，自动更新云容器引擎（CCE）中使用该镜像部署的应用。

图 2-1 SWR 使用流程



3 容器引擎基础知识

容器引擎（即Docker）是一个开源的引擎，可以轻松地为任何应用创建一个轻量级的、可移植的、自给自足的容器。容器镜像服务兼容原生Docker，支持使用Docker CLI和API管理容器镜像。

安装 Docker

在安装Docker前，请了解Docker的基础知识，具体请参见[Docker Documentation](#)。

Docker几乎支持在所有操作系统上安装，用户可以根据需要选择要安装的Docker版本，具体请参见<https://docs.docker.com/engine/install/>。

📖 说明

- 容器镜像的存储可以使用容器镜像服务（SWR），由于SWR支持Docker 1.11.2及以上版本上传镜像，建议下载对应版本。
- 安装Docker需要连接互联网，内网服务器需要绑定弹性IP后才能访问。

另外，在Linux操作系统下，可以使用如下命令快速安装Docker。

```
curl -fsSL get.docker.com -o get-docker.sh
sh get-docker.sh
sudo systemctl daemon-reload
sudo systemctl restart docker
```

制作容器镜像

本节指导您通过Dockerfile定制一个简单的Web应用程序的容器镜像。Dockerfile是一个文本文件，其内包含了一条条的指令（Instruction），每一条指令构建一层，因此每一条指令的内容，就是描述该层应当如何构建。

使用Nginx镜像创建容器应用，在浏览器访问时则会看到默认的Nginx欢迎页面，本节以Nginx镜像为例，修改Nginx镜像的欢迎页面，定制一个新的镜像，将欢迎页面改为“Hello, SWR!”。

步骤1 以root用户登录Docker所在机器。

步骤2 创建一个名为Dockerfile的文件。

```
mkdir mynginx
```

```
cd mynginx
```

touch Dockerfile

步骤3 编辑Dockerfile。

vim Dockerfile

增加文件内容如下：

```
FROM nginx
RUN echo '<h1>Hello,SWR!</h1>' > /usr/share/nginx/html/index.html
```

Dockerfile指令介绍如下。

- FROM语句：表示使用nginx镜像作为基础镜像，一个Dockerfile中FROM是必备的指令，并且必须是第一条指令。
- RUN语句：格式为RUN <命令>，表示执行echo命令，在显示器中显示一段“Hello, SWR!”的文字。

保存并退出。

步骤4 使用docker build [选项] <上下文路径> 构建镜像。

docker build -t nginx:v1 .

- -t nginx:v1：指定镜像的名称和版本。
- .：指定Dockerfile所在目录，镜像构建命令将该路径下所有的内容打包给Docker帮助构建镜像。

步骤5 执行以下命令，可查看到已成功部署的nginx镜像，版本为v1。

docker images

----结束

制作镜像压缩包

本节指导您将容器镜像制作成tar或tar.gz文件压缩包。

步骤1 以root用户登录Docker所在机器。

步骤2 执行如下命令查看镜像。

docker images

查看需要导出的镜像及tag。

步骤3 执行如下命令制作镜像压缩包。

docker save [OPTIONS] IMAGE [IMAGE...]

📖 说明

OPTIONS: --output或-o，表示导出到文件。

压缩包格式为：.tar或.tar.gz。

示例：

```
$ docker save nginx:latest > nginx.tar
$ ls -sh nginx.tar
108M nginx.tar
$ docker save php:5-apache > php.tar.gz
```

```
$ ls -sh php.tar.gz
372M php.tar.gz

$ docker save --output nginx.tar nginx
$ ls -sh nginx.tar
108M nginx.tar

$ docker save -o nginx-all.tar nginx
$ docker save -o nginx-latest.tar nginx:latest
```

---结束

4 镜像管理

4.1 客户端上传镜像

操作场景

本章节以[容器引擎基础知识](#)中制作的nginx:v1镜像为例，介绍如何使用客户端上传镜像。客户端上传镜像，是指在安装了容器引擎的客户端上使用docker命令将镜像上传到容器镜像服务的镜像仓库。

说明

使用客户端上传镜像，镜像的每个layer大小不能超过10G。

约束与限制

- 使用客户端上传镜像，镜像的每个layer大小不能超过10G。
- 上传镜像的容器引擎客户端版本必须为1.11.2及以上。

前提条件

已创建组织，请参见[创建组织](#)。

操作步骤

步骤1 连接容器镜像服务。

1. 登录容器镜像服务控制台，以root用户登录容器引擎所在的虚拟机。
2. 选择容器镜像服务控制台左侧导航栏的“总览”，单击页面右上角的“登录指令”，在弹出的页面中单击  复制登录指令。

说明

- 此处生成的登录指令有效期为24小时，若需要长期有效的登录指令，请参见[获取长期有效登录指令](#)。获取了长期有效的登录指令后，在有效期内的临时登录指令仍然可以使用。
- 登录指令末尾的域名为镜像仓库地址，请记录该地址，后面会使用到。

3. 在安装Docker的机器中执行上一步复制的登录指令。
登录成功会显示“Login Succeeded”。

步骤2 在安装Docker的机器上执行以下命令，为nginx镜像打标签。

```
docker tag [镜像名称1:版本名称1] [镜像仓库地址]/[组织名称]/[镜像名称2:版本名称2]
```

其中，

- [镜像名称1:版本名称1]：请替换为您所要上传的实际镜像的名称和版本名称。
- [镜像仓库地址]：可在SWR控制台上查询，即[步骤1.2](#)中登录指令末尾的域名。
- [组织名称]：请替换为您创建的组织。
- [镜像名称2:版本名称2]：请替换为您期待的镜像名称和镜像版本。

示例：

```
docker tag nginx:v1 swr.ae-ad-1.myhuaweicloud.com/group/nginx:v1
```

步骤3 上传镜像至镜像仓库。

```
docker push [镜像仓库地址]/[组织名称]/[镜像名称2:版本名称2]
```

示例：

```
docker push swr.ae-ad-1.myhuaweicloud.com/group/nginx:v1
```

终端显示如下信息，表明上传镜像成功。

```
6d6b9812c8ae: Pushed
695da0025de6: Pushed
fe4c16cbf7a4: Pushed
v1: digest: sha256:eb7e3bbd8e3040efa71d9c2cacfa12a8e39c6b2ccd15eac12bdc49e0b66cee63 size: 948
```

返回容器镜像服务控制台，在“我的镜像”页面，执行刷新操作后可查看到对应的镜像信息。

----结束

4.2 获取长期有效登录指令

操作场景

本章节介绍如何获取长期有效的登录指令，长期有效登录指令的有效期为永久。

📖 说明

为保证安全，获取登录指令过程建议在开发环境执行。

操作步骤

步骤1 获取区域项目名称、镜像仓库地址。

1. 登录管理控制台，单击右上角您的用户名处，单击“我的凭证”。
2. 在“项目列表”页签中查找当前区域对应的项目。
3. 参考[步骤1.2](#)获取镜像仓库地址，登录指令末尾的域名即为镜像仓库地址。

步骤2 获取AK/SK访问密钥。

📖 说明

访问密钥即AK/SK (Access Key ID/Secret Access Key)，表示一组密钥对，用于验证调用API发起请求的访问者身份，与密码的功能相似。如果您已有AK/SK，可以直接使用，无需再次获取。

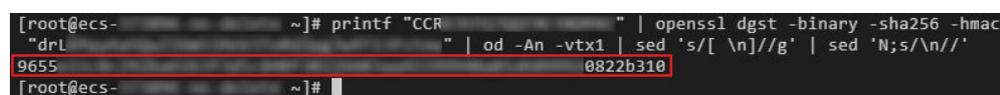
具体操作步骤，请参考[如何获取“中东-阿布扎比-OP5”区域的访问密钥AK/SK](#)

步骤3 登录一台linux系统的计算机，执行如下命令获取登录密钥。

```
printf "$AK" | openssl dgst -binary -sha256 -hmac "$SK" | od -An -vtx1 | sed 's/[ \n]//g' | sed 'N;s/\n/'
```

其中\$AK和\$SK为[步骤2](#)获取的AK/SK。

图 4-1 示例



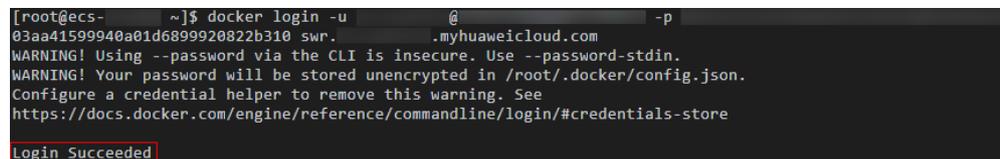
```
[root@ecs- ~]# printf "CCR" " " | openssl dgst -binary -sha256 -hmac
"drL" " " | od -An -vtx1 | sed 's/[ \n]//g' | sed 'N;s/\n/'
9655                                0822b310
[root@ecs- ~]#
```

步骤4 使用如下的格式拼接登录指令。

```
docker login -u [区域项目名]@[AK] -p [登录密钥] [镜像仓库地址]
```

其中，区域项目名和镜像仓库地址在[步骤1](#)中获取，AK在[步骤2](#)中获取，登录密钥为[步骤3](#)的执行结果。

图 4-2 长期登录指令



```
[root@ecs- ~]$ docker login -u @ -p
03aa41599940a01d6899920822b310 swr. .myhuaweicloud.com
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

📖 说明

登录密钥字符串是经过加密的，无法逆向解密，从-p无法获取到SK。

获取的登录指令可在其他机器上使用并登录。

步骤5 使用history -c命令清理相关使用痕迹，避免隐私信息泄露。

----结束

4.3 页面上传镜像

操作场景

本章节介绍如何通过页面上传镜像。从页面上传镜像，是指直接通过控制台页面将镜像上传到容器镜像服务的镜像仓库。

约束与限制

- 每次最多上传10个文件，单个文件大小（含解压后）不得超过2G。

- 仅支持上传1.11.2及以上容器引擎客户端版本制作的镜像压缩包。

前提条件

- 已创建组织，请参见[创建组织](#)。
- 镜像已存为tar或tar.gz文件，具体请参见[制作镜像压缩包](#)。

操作步骤

步骤1 登录容器镜像服务控制台。

步骤2 在左侧导航栏选择“我的镜像”，单击右上角“页面上传”。

步骤3 在弹出的窗口中选择组织，单击“选择镜像文件”，选择要上传的镜像文件。

说明

多个镜像同时上传时，镜像文件会按照顺序逐个上传，不支持并发上传。

图 4-3 上传镜像



步骤4 单击“开始上传”。

待任务进度显示“上传完成”，表示镜像上传成功。

---结束

4.4 下载镜像

操作场景

您可以使用docker pull命令下载容器镜像服务中的镜像。

操作步骤

- 步骤1** 以root用户登录容器引擎所在的虚拟机。
- 步骤2** 参考**步骤1**获取登录访问权限，连接容器镜像服务。
- 步骤3** 登录容器镜像服务控制台。
- 步骤4** 在左侧导航栏选择“我的镜像”，单击右侧镜像名称。
- 步骤5** 在镜像详情页面中，单击对应镜像版本“下载指令”列的复制图标，复制镜像下载指令。

图 4-4 获取镜像下载指令



- 步骤6** 在虚拟机中执行**步骤5**复制的镜像下载指令。

使用**docker images**命令查看是否下载成功。

```
# docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
xxx/group/nginx     v2.0.0     22f2bf2e2b4f 5 hours ago   22.8MB
```

- 步骤7** （可选）执行如下命令将镜像保存为归档文件。

```
docker save [镜像名称:版本名称] > [归档文件名称]
```

----结束

4.5 编辑镜像属性

操作场景

镜像上传后默认为私有镜像，您可以设置镜像的属性，包括镜像的类型（“公开”或“私有”）、分类及描述。

公开镜像所有用户都能下载，私有镜像则受具体权限管理控制。您可以为用户添加授权，授权完成后，用户享有读取、编辑或管理私有镜像的权限，具体请参见[在镜像详情中添加授权](#)。

操作步骤

- 步骤1** 登录容器镜像服务控制台。
- 步骤2** 在左侧导航栏选择“我的镜像”，单击右侧要编辑镜像的名称。
- 步骤3** 在镜像详情页面，单击右上角“编辑”，在弹出的窗口中根据需要编辑类型（“公开”或“私有”）、分类及描述，然后单击“确定”。

图 4-5 编辑镜像属性

编辑镜像

×

组织 cce

名称 tomcat

类型
 公开
 私有

分类

描述

确定
取消

表 4-1 编辑镜像

| 参数 | 说明 |
|----|---|
| 组织 | 镜像所属组织。 |
| 名称 | 镜像名称。 |
| 类型 | 镜像类型，可选择： <ul style="list-style-type: none"> ● 公开 ● 私有 说明 公开镜像所有用户都可以下载使用。 <ul style="list-style-type: none"> ● 如果您的机器与镜像仓库在同一区域，访问仓库是通过内网访问。 ● 如果您的机器与镜像仓库在不同区域，通过公网才能访问仓库，下载跨区域仓库的镜像需要节点可以访问公网。 |
| 分类 | 镜像分类，可选择： <ul style="list-style-type: none"> ● 应用服务器 ● Linux ● Windows ● Arm ● 框架与应用 ● 数据库 ● 语言 ● 其他 |
| 描述 | 输入镜像仓库描述，0-30000个字符。 |

----结束

4.6 共享私有镜像

操作场景

镜像上传后，您可以共享**私有镜像**给其他帐号，并授予下载该镜像的权限。

被共享的用户需要登录容器镜像服务控制台，在“我的镜像 > 他人共享”页面查看共享的镜像。被共享的用户单击镜像名称，可进入镜像详情页面查看镜像版本、下载指令等。

约束与限制

- 镜像共享功能只支持私有镜像进行共享，不支持公有镜像共享。
- 仅具备该私有镜像管理权限的用户才能共享镜像，被共享者只有只读权限，只能下载镜像。
- 镜像共享功能只能在同一区域内使用，不支持在不同区域间镜像共享。

操作步骤

- 步骤1** 登录容器镜像服务控制台。
- 步骤2** 在左侧导航栏选择“我的镜像”，单击右侧镜像的名称。
- 步骤3** 在镜像详情页面选择“共享”页签。
- 步骤4** 单击“共享镜像”，根据[表4-2](#)填写相关参数，然后单击“确定”。

图 4-6 共享镜像

共享镜像

账户输入错误次数过多，共享镜像操作将会被限制，请您谨慎操作

* 共享给 请输入账号名称

* 截止日期 2021/09/09 X 永久有效

描述 请输入不超过1000个字符。 0/1,000

* 权限 下载

确定 取消

表 4-2 共享镜像

| 参数 | 说明 |
|------|-----------------------------|
| 共享给 | 输入帐号名称。 |
| 截止日期 | 选择共享截止日期。如勾选“永久有效”，则共享永久有效。 |
| 描述 | 输入描述，0-1000个字符。 |
| 权限 | 当前仅支持“下载”权限。 |

步骤5 共享完成后，您可以在“我的镜像 > 自有镜像”中，勾选“我共享的镜像”，查看所有共享的镜像。

----结束

4.7 添加触发器

操作场景

容器镜像服务可搭配云容器引擎CCE一起使用，实现镜像版本更新时自动更新使用该镜像的应用。您只需要为镜像添加一个触发器。通过触发器，可以在每次生成新的镜像版本时，自动执行更新动作，如：自动更新使用该镜像的应用。

前提条件

更新应用镜像版本之前，请确保已创建容器应用，将镜像部署到云容器引擎CCE。
如未创建，请登录云容器引擎工作负载页面进行创建。

操作步骤

- 步骤1** 登录容器镜像服务控制台。
- 步骤2** 在左侧导航栏选择“我的镜像”，单击右侧镜像名称，进入镜像详情页。
- 步骤3** 选择“触发器”页签，单击“添加触发器”，根据表4-3填写相关参数，然后单击“确定”。

图 4-7 添加触发器

添加触发器

通过触发器，可以在每次生成新的镜像版本时，自动执行更新动作，如：自动更新使用该镜像的应用

触发器名称 字母开头，由字母、数字、下划线(_)、中划线(-)组成，下划线、中划线不能连续且不能作为结尾，1-64个字符。

触发条件

触发动作 需指定更新的应用,以及该应用下的指定容器镜像

触发器状态 启用 停用

触发器类型

| 集群 | 命名空间 | 应用 | 容器 |
|-----------------------------------|-----------------------------------|---------------------------------|---------------------------------|
| <input type="text" value="选择应用"/> | <input type="text" value="命名空间"/> | <input type="text" value="应用"/> | <input type="text" value="容器"/> |

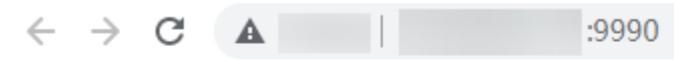
表 4-3 触发器

| 参数 | 说明 |
|-------|---|
| 触发器名称 | 字母开头，由字母、数字、下划线_、中划线-组成，下划线、中划线不能连续且不能作为结尾，1-64个字符。 |
| 触发条件 | 支持如下三种触发条件，当镜像有新版本时，触发部署应用。 <ul style="list-style-type: none">● 全部触发：有新的镜像版本生成或镜像版本发生更新时，触发部署。● 指定版本号触发：有指定镜像版本生成或更新时，触发部署。● 正则触发：有符合正则表达式的镜像版本生成或更新时，触发部署。正则表达式规则如下：<ul style="list-style-type: none">- *：匹配不包含路径分隔符“/”的任何字段。- **：匹配包含路径分隔符“/”的任何字段。- ?：匹配任何单个非“/”的字符。- {选项1, 选项2, ...}：同时匹配多个选项。 |
| 触发动作 | 当前仅支持更新容器的镜像，需指定更新的应用，以及该应用下的指定容器镜像。 |
| 触发器状态 | 选择“启用”。 |
| 触发器类型 | 选择“云容器引擎CCE”。 |
| 选择应用 | 选择要更新镜像的容器。 |

----结束

示例

假设有一个欢迎页面为“Hello, SWR!”的Nginx镜像（版本号为v1），使用该镜像创建了名称为“nginx”的无状态负载，该负载提供对外访问。

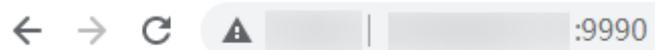


Hello, SWR!

1. 为Nginx镜像添加触发器。
触发器名称填写“All_tags”，触发条件选择“全部触发”，选择使用了Nginx镜像的无状态负载及容器。
2. Nginx镜像新增一个v2版本，该版本的欢迎页面为“Hello, SoftWare Repository for Container!”。
3. 确认是否触发成功。

在“触发器”页签，单击图标，查看触发结果为“成功”。

工作负载的访问页面已变更为“Hello, SoftWare Repository for Container!”。



Hello, SoftWare Repository for Container!

5 组织管理

操作场景

组织用于隔离镜像仓库，每个组织可对应一个公司或部门，将其拥有的镜像集中在该组织下。在不同的组织下，可以有同名的镜像。同一用户可属于不同的组织，如图5-1所示。

SWR支持为用户分配相应的访问权限（读取、编辑、管理），具体请参见[授权管理](#)。

图 5-1 组织



创建组织

容器镜像服务为您提供组织管理功能，方便您根据自身组织架构来构建镜像的资源管理。上传镜像前，请先创建组织。

步骤1 登录容器镜像服务控制台。

步骤2 在左侧导航栏选择“组织管理”，单击右上角“创建组织”，在弹出的页面中填写“组织名称”，然后单击“确定”。

图 5-2 创建组织



说明

- 组织名称全局唯一，创建组织时如果提示组织已存在，可能该组织名称已被其他用户使用，请重新设置一个组织名称。
- 创建组织时，如系统提示组织已存在，可能是删除租户后，组织资源存在残留，建议您重新设置一个组织名称。

---结束

查看组织中的镜像

创建组织后，您可以查看当前组织中的镜像。

步骤1 登录容器镜像服务控制台。

步骤2 在左侧导航栏选择“组织管理”，单击右侧组织名称。

步骤3 单击“镜像”页签，查看当前组织中的镜像。

图 5-3 查看组织中的镜像



| 镜像名称 | 版本数 | 更新时间 |
|-------|-----|-------------------------------|
| nginx | 1 | 2021/06/19 23:21:04 GMT+08:00 |

----结束

删除组织

删除组织前，请先删除组织下的所有镜像。

- 步骤1** 登录容器镜像服务控制台。
- 步骤2** 在左侧导航栏选择“组织管理”，单击右侧组织名称。
- 步骤3** 单击右上角“删除”按钮，在弹出的对话框中根据提示输入“DELETE”，然后单击“确定”。

----结束

须知

如果需要删除租户，需要先删除组织，直接删除租户会导致其下组织无法清理，否则再次创建同名组织时系统会提示已存在。

6 授权管理

操作场景

如果您需要对容器镜像服务进行权限管理，您可以使用统一身份认证服务IAM。当您具有SWR Admin或者Tenant Administrator系统权限时，您就拥有了SWR的管理员权限，可以在SWR中为其他IAM用户进行授权。

说明

拥有SWR管理员权限的用户，默认拥有所有组织下的镜像管理权限，即使该用户不在组织的授权用户列表中。

如果您没有SWR的管理员权限，就需要已拥有SWR管理员权限的用户在SWR中进行授权管理，为您添加对某个镜像的权限或对某个组织中所有镜像的权限。

授权方法

容器镜像服务中给用户添加权限有如下两种方法：

- [在镜像详情中添加授权](#)，授权完成后，用户享有读取/编辑/管理该镜像的权限。
- [在组织中添加授权](#)，使用户对组织内所有镜像享有读取/编辑/管理的权限。

图 6-1 用户权限



容器镜像服务中为用户添加的权限有如下三种类型：

- 读取：只能下载镜像，不能上传。
- 编辑：下载镜像、上传镜像、编辑镜像属性以及创建触发器。
- 管理：下载镜像、上传镜像、删除镜像或版本、编辑镜像属性、添加授权、添加触发器以及共享镜像。

📖 说明

页面上传镜像功能要求具备组织的编辑或管理权限，在镜像详情中添加的编辑或管理权限不支持页面上传镜像。

在镜像详情中添加授权

在镜像详情中为用户添加授权，授权完成后，该用户享有读取/编辑/管理该镜像的权限。

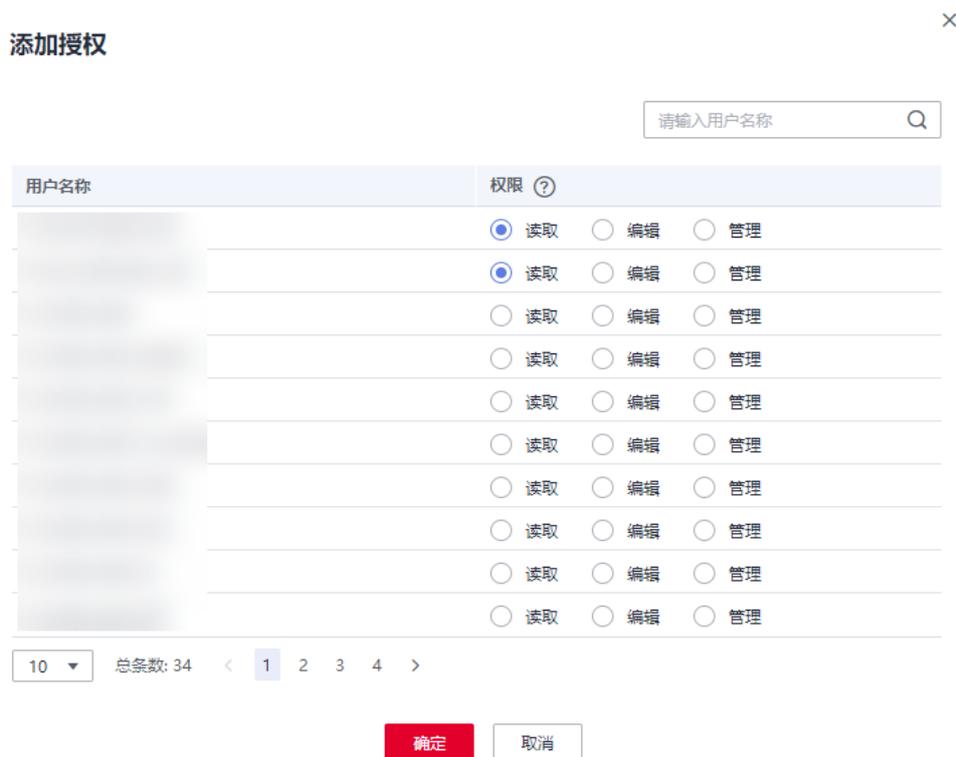
步骤1 登录容器镜像服务控制台。

步骤2 在左侧导航栏选择“我的镜像”，单击右侧要编辑镜像的名称。

步骤3 在镜像详情页面选择“权限管理”页签。

步骤4 单击“添加授权”，选择用户名称，添加“读取/编辑/管理”的权限，添加后，该用户享有对应权限。

图 6-2 在镜像详情中添加授权



----结束

在镜像详情中修改/删除授权

您还可以在镜像详情中修改用户权限及删除用户权限。

- 修改授权：在“权限管理”页签下用户所在行单击“修改”，在“权限”所在列选择新的权限，然后单击“保存”。
- 删除授权：在“权限管理”页签下用户所在行单击“删除”。在弹出的对话框中根据提示输入“DELETE”，然后单击“确定”。

在组织中添加授权

在组织中为用户添加授权，使用户对组织内所有镜像享有读取/编辑/管理的权限。

只有具备“管理”权限的用户才能添加授权。

步骤1 登录容器镜像服务控制台。

步骤2 在左侧导航栏选择“组织管理”，单击右侧组织名称后的“详情”。

步骤3 在“用户”页签下单击“添加授权”，在弹出的窗口中为用户选择权限，然后单击“确定”。

----结束

在组织中修改/删除授权

您还可以在组织中修改用户权限及删除用户权限。

- 修改授权：在“用户”页签下用户所在行单击“修改”，在“权限”所在列选择新的权限，然后单击“保存”。
- 在“用户”页签下用户所在行单击“删除”。在弹出的对话框中根据提示输入“DELETE”，然后单击“确定”。

7 审计

7.1 支持云审计的关键操作

操作场景

云审计服务（Cloud Trace Service, CTS），是云安全解决方案中专业的日志审计服务，提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

通过云审计服务，您可以记录与SWR相关的操作事件，便于日后的查询、审计和回溯。

支持审计的关键操作列表

表 7-1 云审计服务支持的 SWR 操作列表

| 操作名称 | 资源类型 | 事件名称 |
|----------|-------------------|-------------------------|
| 创建命名空间权限 | usernamespaceauth | createUserNamespaceAuth |
| 修改命名空间权限 | usernamespaceauth | updateUserNamespaceAuth |
| 删除命名空间权限 | usernamespaceauth | deleteUserNamespaceAuth |
| 创建软件包 | package | createPackage |
| 修改软件包 | package | updatePackage |
| 删除软件包 | package | deletePackage |
| 创建仓库 | repository | createRepository |
| 修改仓库 | repository | updateRepository |
| 删除仓库 | repository | deleteRepository |
| 创建版本 | version | createVersion |
| 修改版本 | version | updateVersion |

| 操作名称 | 资源类型 | 事件名称 |
|--------|---------------------------------|---------------------------------------|
| 删除版本 | version | deleteVersion |
| 上传镜像包 | image | uploadImagePackage |
| 上传文件 | file | uploadFile |
| 下载文件 | file | downloadFile |
| 删除文件 | file | deleteFile |
| 创建组织 | usernamepace | createUserNamespace |
| 删除组织 | usernamepace | deleteUserNamesapce |
| 创建触发器 | trigger | createTrigger |
| 修改触发器 | trigger | updateTrigger |
| 删除触发器 | trigger | deleteTrigger |
| 创建仓库权限 | userrepositoryauth | createUserRepositoryAuth |
| 修改仓库权限 | userrepositoryauth | updateUserRepositoryAuth |
| 删除仓库权限 | userrepositoryauth | deleteUserRepositoryAuth |
| 创建镜像仓库 | imagerepository | createImageRepository |
| 修改镜像仓库 | imagerepository | updateImageRepository |
| 删除镜像仓库 | imagerepository | deleteImageRepository |
| 删除镜像版本 | imagetag | deleteImageTag |
| 生成登录指令 | dockerlogincmd | createDockerConfig |
| 创建共享镜像 | imagerepositoryaccessd omain | createImageRepositoryAccess Domain |
| 修改共享镜像 | imagerepositoryaccessd omain | updateImageRepositoryAccess Domain |
| 删除共享镜像 | imagerepositoryaccessd omain | deleteImageRepositoryAccess Domain |

7.2 查看云审计日志

操作场景

开启了云审计服务（CTS）后，系统开始记录SWR相关的操作。CTS会保存最近1周的操作记录。

本小节介绍如何在CTS管理控制台查看最近1周的操作记录。

操作步骤

步骤1 登录CTS管理控制台。

步骤2 选择左侧导航栏的“事件列表”，进入事件列表页面。

步骤3 事件记录了云资源的操作详情，设置筛选条件，单击“查询”。

当前事件列表支持四个维度的组合查询，详细信息如下：

- 事件类型、事件来源、资源类型和筛选类型。

在下拉框中选择查询条件。其中，“事件类型”选择“管理事件”，“事件来源”选择“SWR”。

图 7-1 设置筛选条件



其中，

- 筛选类型选择“按资源ID”时，还需手动输入某个具体的资源ID，目前仅支持全字匹配模式的查询。
- 筛选类型选择“按资源名称”时，还需选择或手动输入某个具体的资源名称。
- 操作用户：在下拉框中选择某一具体的操作用户。
- 事件级别：可选项为“所有事件级别”、“Normal”、“Warning”、“Incident”，只可选择其中一项。
- 时间范围：可选项为“最近1小时”、“最近1天”、“最近1周”和“自定义时间段”，本示例选择“最近1周”。

步骤4 在需要查看的事件左侧，单击  图标展开该事件的详细信息。

步骤5 在需要查看的事件右侧，单击“查看事件”，弹出一个窗口，显示了该操作事件结构的详细信息。

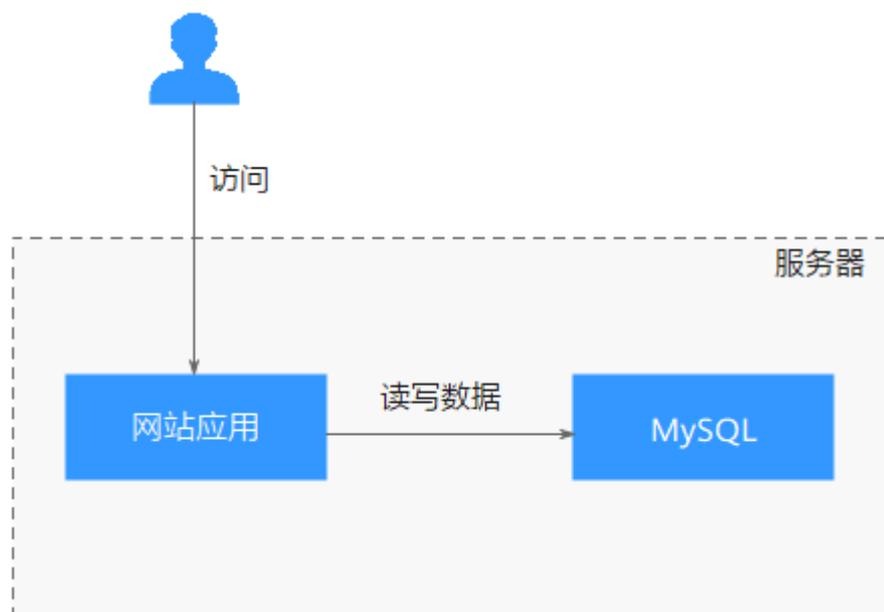
----结束

8 最佳实践

8.1 编写高效的 Dockerfile

本章基于容器镜像服务实践所编写，将一个单体应用进行容器改造为例，展示如何写出可读性更好的Dockerfile，从而提升镜像构建速度，构建层数更少、体积更小的镜像。

下面是一个常见企业门户网站架构，由一个Web Server和一个数据库组成，Web Server提供Web服务，数据库保存用户数据。通常情况下，这样一个门户网站安装在一台服务器上。



如果把应用运行在一个Docker容器中，那么很可能写出下面这样的Dockerfile来。

```
FROM ubuntu
ADD . /app

RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y nodejs ssh mysql
```

```
RUN cd /app && npm install

# this should start three processes, mysql and ssh
# in the background and node app in foreground
# isn't it beautifully terrible? <3
CMD mysql & sshd & npm start
```

当然这样Dockerfile有很多问题，这里CMD命令是错误的，只是为了说明问题而写。

下面的内容中将展示对这个Dockerfile进行改造，说明如何写出更好的Dockerfile，共有如下几种处理方法。

- 一个容器只运行一个进程
- 不要在构建中升级版本
- 将变化频率一样的RUN指令合一
- 使用特定的标签
- 删除多余文件
- 选择合适的基础镜像
- 设置WORKDIR和CMD
- 使用ENTRYPOINT（可选）
- ENTRYPOINT脚本中使用exec
- 优先使用COPY
- 合理调整COPY与RUN的顺序
- 设置默认的环境变量、映射端口和数据卷
- 使用EXPOSE暴露端口
- 使用VOLUME管理数据卷
- 使用LABEL设置镜像元数据
- 添加HEALTHCHECK
- 编写.dockerignore文件

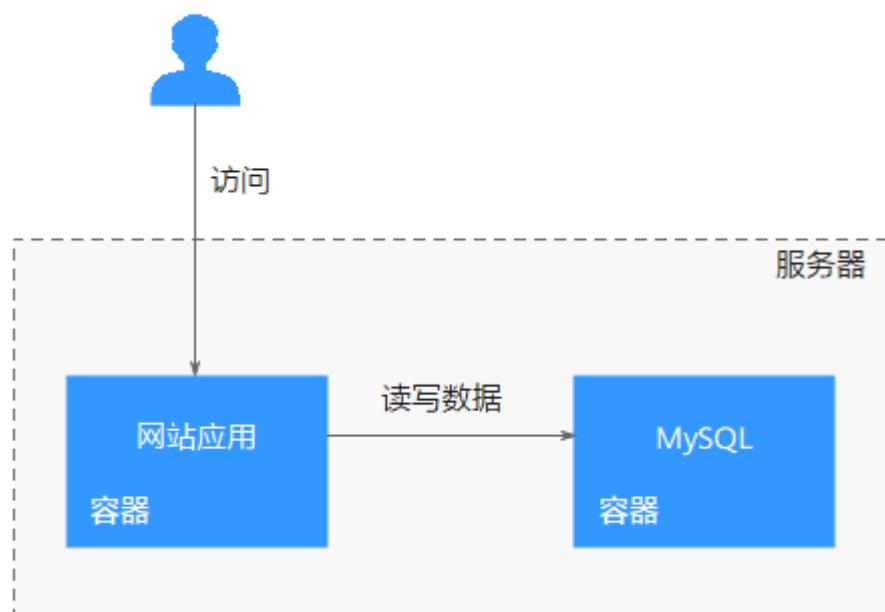
一个容器只运行一个进程

从技术角度讲，Docker容器中可以运行多个进程，您可以将数据库、前端、后端、ssh等都运行在同一个Docker容器中。但是，这样跟未使用容器前没有太大区别，且这样容器的构建时间非常长（一处修改就要构建全部），镜像体积大，横向扩展时非常浪费资源（不同的应用需要运行的容器数并不相同）。

通常所说的容器化改造是对应用整体微服务进行架构改造，改造后，再容器化。这样做可以带来如下好处：

- 单独扩展：拆分为微服务后，可单独增加或缩减每个微服务的实例数量。
- 提升开发速度：各微服务之间解耦，某个微服务的代码开发不影响其他微服务。
- 通过隔离确保安全：整体应用中，若存在安全漏洞，一旦被攻击，所有功能的权限都可能会被窃取。微服务架构中，若攻击了某个服务，只可获得该服务的访问权限，无法入侵其他服务。
- 提升稳定性：如果其中一个微服务崩溃，其他微服务还可以持续正常运行。

因此，上述企业门户网站可以进行如下改造，Web应用和MySQL运行在不同容器中。



MySQL运行在独立的镜像中，这样的好处就是，我们可以对它们分别进行修改，且不会牵一发而动全身。如下面这个例子所示，我们可以删除MySQL，只安装node.js。

```
FROM ubuntu
ADD ./app
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y nodejs
RUN cd /app && npm install
CMD npm start
```

不要在构建中升级版本

为了降低复杂性、减少依赖、减小文件大小、节约构建时间，你应该避免安装任何不必要的包。例如，不要在数据库镜像中包含一个文本编辑器。

如果基础镜像中的某个包过时了，但你不知道具体是哪一个包，你应该联系它的维护者。如果你确定某个特定的包，比如foo需要升级，使用`apt-get install -y foo`就行，该指令会自动升级foo包。

`apt-get upgrade`会使得镜像构建过程非常不稳定，在构建时不确定哪些包会被安装，此时可能会产生不一致的镜像。因此通常会删掉`apt-get upgrade`。

删掉`apt-get upgrade`后，Dockerfile如下：

```
FROM ubuntu
ADD ./app
RUN apt-get update
RUN apt-get install -y nodejs
RUN cd /app && npm install
CMD npm start
```

将变化频率一样的 RUN 指令合一

Docker镜像是分层的，类似于洋葱，它们都有很多层，为了修改内层，则需要将外面的层都删掉。Docker镜像有如下特性：

- Dockerfile中的每个指令都会创建一个新的镜像层。
- 镜像层将被缓存和复用。
- Dockerfile修改后，复制的文件变化了或者构建镜像时指定的变量不同了，对应的镜像层缓存就会失效。
- 某一层的镜像缓存失效之后，它之后的镜像层缓存都会失效。
- 镜像层是不可变的，如果我们在某一层中添加一个文件，然后在下一层中删除它，则镜像中依然会包含该文件，只是这个文件在Docker容器中不可见。

将变化频率一样的指令合并在一起，目的是为了更好的将镜像分层，避免带来不必要的成本。如本例中将node.js安装与npm模块安装放在一起的话，则每次修改源代码，都需要重新安装node.js，这显然不合适。

```
FROM ubuntu
ADD ./app
RUN apt-get update \
    && apt-get install -y nodejs \
    && cd /app \
    && npm install
CMD npm start
```

因此，正确的写法是这样的：

```
FROM ubuntu
RUN apt-get update && apt-get install -y nodejs
ADD ./app
RUN cd /app && npm install
CMD npm start
```

使用特定的标签

当镜像没有指定标签时，将默认使用latest标签。因此，FROM ubuntu指令等同于FROM ubuntu:latest。当镜像更新时，latest标签会指向不同的镜像，这时构建镜像有可能失败。

如下示例中使用16.04作为标签。

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y nodejs
ADD ./app
RUN cd /app && npm install
CMD npm start
```

删除多余文件

假设我们更新了apt-get源，下载解压并安装了一些软件包，它们都保存在“/var/lib/apt/lists/”目录中。

但是，运行应用时Docker镜像中并不需要这些文件。因此最好将它们删除，因为它会使Docker镜像变大。

示例Dockerfile中，删除“/var/lib/apt/lists/”目录中的文件。

```
FROM ubuntu:16.04

RUN apt-get update \
  && apt-get install -y nodejs \
  && rm -rf /var/lib/apt/lists/*

ADD ./app
RUN cd /app && npm install

CMD npm start
```

选择合适的基础镜像

在示例中，我们选择了ubuntu作为基础镜像。但是我们只需要运行node程序，没有必要使用一个通用的基础镜像，node镜像应该是更好的选择。

更好的选择是alpine版本的node镜像。alpine是一个极小化的Linux发行版，只有4MB，这让它非常适合作为基础镜像。

```
FROM node:7-alpine

ADD ./app
RUN cd /app && npm install

CMD npm start
```

设置 WORKDIR 和 CMD

WORKDIR指令可以设置默认目录，也就是运行RUN / CMD / ENTRYPOINT指令的地方。

CMD指令可以设置容器创建时执行的默认命令。另外，您应该将命令写在一个数组中，数组中每个元素为命令的每个单词。

```
FROM node:7-alpine

WORKDIR /app
ADD ./app
RUN npm install

CMD ["npm", "start"]
```

使用 ENTRYPOINT（可选）

ENTRYPOINT指令并不是必须的，因为它会增加复杂度。ENTRYPOINT是一个脚本，它会默认执行，并且将指定的命令作为其参数。它通常用于构建可执行的Docker镜像。

```
FROM node:7-alpine

WORKDIR /app
ADD ./app
RUN npm install

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

ENTRYPOINT 脚本中使用 exec

在前文的ENTRYPOINT脚本中，使用了exec命令运行node应用。不使用exec的话，我们则不能顺利地关闭容器，因为SIGTERM信号会被bash脚本进程吞没。exec命令启动的进程可以取代脚本进程，因此所有的信号都会正常工作。

优先使用 COPY

COPY指令非常简单，仅用于将文件拷贝到镜像中。ADD相对来讲复杂一些，可以用于下载远程文件以及解压压缩包。

```
FROM node:7-alpine
WORKDIR /app
COPY . /app
RUN npm install

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

合理调整 COPY 与 RUN 的顺序

将变化最少的部分放在Dockerfile的前面，这样可以充分利用镜像缓存。

示例中，源代码会经常变化，则每次构建镜像时都需要重新安装NPM模块，这显然不是我们希望看到的。因此我们可以先拷贝package.json，然后安装NPM模块，最后才拷贝其余的源代码。这样的话，即使源代码变化，也不需要重新安装NPM模块。

```
FROM node:7-alpine
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

设置默认的环境变量、映射端口和数据卷

运行Docker容器时很可能需要一些环境变量。在Dockerfile设置默认的环境变量是一种很好的方式。另外，我们应该在Dockerfile中设置映射端口和数据卷。示例如下：

```
FROM node:7-alpine
ENV PROJECT_DIR=/app
WORKDIR $PROJECT_DIR
COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

ENV指令指定的环境变量在容器中可以正常使用。如果你只是需要指定构建镜像时的变量，你可以使用ARG指令。

使用 EXPOSE 暴露端口

EXPOSE指令用于指定容器将要监听的端口。因此，你应该为你的应用程序使用常见的端口。例如，提供Apache web服务的镜像应该使用EXPOSE 80，而提供MongoDB服务的镜像使用EXPOSE 27017。

对于外部访问，用户可以在执行docker run时使用一个标志来指示如何将指定的端口映射到所选择的端口。

```
FROM node:7-alpine

ENV PROJECT_DIR=/app

WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV APP_PORT=3000
EXPOSE $APP_PORT

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

使用 VOLUME 管理数据卷

VOLUME指令用于暴露任何数据库存储文件、配置文件或容器创建的文件和目录。强烈建议使用VOLUME来管理镜像中的可变部分和用户可以改变的部分。

下面示例中填写一个媒体目录。

```
FROM node:7-alpine

ENV PROJECT_DIR=/app

WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV MEDIA_DIR=/media \
    APP_PORT=3000

VOLUME $MEDIA_DIR
EXPOSE $APP_PORT

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

使用 LABEL 设置镜像元数据

你可以给镜像添加标签来帮助组织镜像、记录许可信息、辅助自动化构建等。每个标签一行，由LABEL开头加上一个或多个标签对。

须知

如果你的字符串中包含空格，必须将字符串放入引号中或者对空格使用转义。如果字符串内容本身就包含引号，必须对引号使用转义。

```
FROM node:7-alpine
LABEL com.example.version="0.0.1-beta"
```

添加 HEALTHCHECK

运行容器时，可以指定`--restart always`选项。这样的话，容器崩溃时，docker daemon会重启容器。对于需要长时间运行的容器，这个选项非常有用。但是，如果容器的确在运行，但是不可用怎么办？使用HEALTHCHECK指令可以让Docker周期性的检查容器的健康状况。我们只需要指定一个命令，如果一切正常的话返回0，否则返回1。当请求失败时，`curl --fail`命令返回非0状态。示例如下：

```
FROM node:7-alpine
LABEL com.example.version="0.0.1-beta"

ENV PROJECT_DIR=/app
WORKDIR $PROJECT_DIR

COPY package.json $PROJECT_DIR
RUN npm install
COPY . $PROJECT_DIR

ENV MEDIA_DIR=/media \
    APP_PORT=3000

VOLUME $MEDIA_DIR
EXPOSE $APP_PORT
HEALTHCHECK CMD curl --fail http://localhost:$APP_PORT || exit 1

ENTRYPOINT ["/entrypoint.sh"]
CMD ["start"]
```

编写.dockerignore 文件

.dockerignore的作用和语法类似于.gitignore，可以忽略一些不需要的文件，这样可以有效加快镜像构建时间，同时减少Docker镜像的大小。

构建镜像时，Docker需要先准备context，将所有需要的文件收集到进程中。默认的context包含Dockerfile目录中的所有文件，但是实际上，我们并不需要.git目录等内容。

示例如下：

```
.git/
```

9 常见问题

9.1 通用类

9.1.1 产品咨询

SWR 最多能存储多少个镜像？

SWR对存储的镜像数量没有限制，您可以根据需要上传镜像。

SWR 镜像仓库支持 ARM 镜像上传吗？

SWR镜像仓库对镜像的内核架构是没有任何限制的，您上传ARM架构的镜像和上传x86的镜像是没有区别的，直接上传即可。

docker push 使用什么协议将镜像推送到 SWR？

HTTPS协议。

同名同 tag 的镜像上传后会覆盖之前的镜像吗？

会覆盖，保留最新上传的镜像。

9.1.2 如何制作容器镜像？

自己制作容器镜像，主要有两种方法：

- 制作快照方式获得镜像（偶尔制作的镜像）：在基础镜像上（比如Ubuntu），先登录镜像系统并安装容器引擎软件，然后整体制作快照。
- Dockerfile方式构建镜像（经常更新的镜像）：将软件安装的流程写成Dockerfile，使用docker build构建容器镜像。

方法一：制作快照方式获得镜像

如果后续镜像没有变化，可采用[方法一](#)制作镜像。



具体操作如下：

1. 找一台主机，安装容器引擎软件。
2. 启动一个空白的基础容器，并进入容器。
例如：启动一个CentOS的容器。

```
docker run -it centos
```

3. 执行安装任务。

```
yum install XXX  
git clone https://github.com/lh3/bwa.git  
cd bwa;make
```

📖 说明

请预先安装好Git，并检查本机是否有ssh key设置。

4. 输入**exit**退出容器。
5. 制作快照。

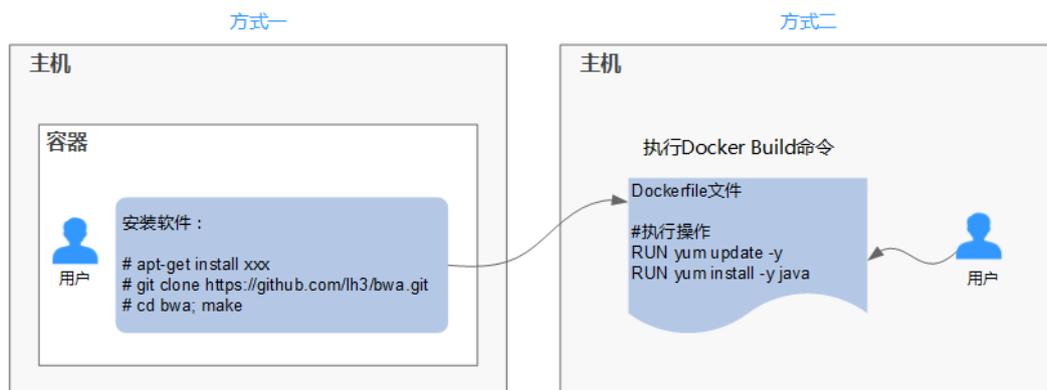
```
docker commit -m "xx" -a "test" container-id test/image:tag
```

 - -a: 提交的镜像作者。
 - container-id: [步骤2](#)中的容器id。可以使用**docker ps -a**查询得到容器id。
 - -m: 提交时的说明文字。
 - test/image:tag: 仓库名/镜像名:TAG名。
6. 执行**docker images**可以查看到制作完成的容器镜像。

方法二：使用 Dockerfile 方式构建

如果后续镜像经常变更（例如某个软件更新版本），则需要采用[方法二](#)制作镜像。若仍采用[方法一](#)制作镜像，则每次变更都需要重新执行[方法一](#)的命令，操作过程比较繁琐，所以建议使用自动化制作镜像的方法。

其实就是将[方法一](#)制作镜像的方法，用文件方式写出来（文件名为Dockerfile）。然后执行：**docker build -t test/image:tag.**命令（命令中“.”表示Dockerfile文件的路径），自动完成镜像制作。



简单的Dockerfile示例：

📖 说明

如果依赖外部网络，请搭建网络环境，并保证网络可用。

```
#Version 1.0.1
FROM centos:latest

#设置root用户为后续命令的执行者
USER root

#执行操作
RUN yum update -y
RUN yum install -y java

#使用&&拼接命令
RUN touch test.txt && echo "abc" >>abc.txt

#对外暴露端口
EXPOSE 80 8080 1038

#添加网络文件
ADD https://www.baidu.com/img/bd_logo1.png /opt/

#设置环境变量
ENV WEBAPP_PORT=9090

#设置工作目录
WORKDIR /opt/

#设置启动命令
ENTRYPOINT ["ls"]

#设置启动参数
CMD ["-a", "-l"]

#设置卷
VOLUME ["/data", "/var/www"]

#设置子镜像的触发操作
ONBUILD ADD . /app/src
ONBUILD RUN echo "on build excuted" >> onbuild.txt
```

Dockerfile 基本语法

- FROM：
指定待扩展的父级镜像（基础镜像）。除注释之外，文件开头必须是一个FROM指令，后面的指令便在这个父级镜像的环境中运行，直到遇到下一个FROM指令。通过添加多个FROM命令，可以在同一个Dockerfile文件中创建多个镜像。

- **MAINTAINER:**
声明创建镜像的作者信息：用户名、邮箱，非必须参数。
- **RUN:**
修改镜像的命令，常用来安装库、安装程序以及配置程序。一条RUN指令执行完毕后，会在当前镜像上创建一个新的镜像层，接下来对的指令会在新的镜像上继续执行。RUN 语句有两种形式：
 - **RUN yum update:** 在/bin/sh路径中执行的指令命令。
 - **RUN ["yum", "update"]:** 直接使用系统调用exec来执行。
 - **RUN yum update && yum install nginx:** 使用&&符号将多条命令连接在同一条RUN语句中。
- **EXPOSE:**
指明容器内进程对外开放的端口，多个端口之间使用空格隔开。
运行容器时，通过设置参数-P（大写）即可将EXPOSE里所指定的端口映射到主机上其他的随机端口，其他容器或主机可以通过映射后的端口与此容器通信。
您也可以通过设置参数-p（小写）将Dockerfile中EXPOSE中没有列出的端口设置成公开。
- **ADD:**
向新镜像中添加文件，这个文件可以是一个主机文件，也可以是一个网络文件，也可以是一个文件夹。
 - 第一个参数：源文件（夹）。
 - 如果是相对路径，必须是相对于Dockerfile所在目录的相对路径。
 - 如果是URL，会将文件先下载下来，然后再添加到镜像里。
 - 第二个参数：目标路径。
 - 如果源文件是主机上的zip或者tar形式的压缩文件，容器引擎会先解压缩，然后将文件添加到镜像的指定位置。
 - 如果源文件是一个通过URL指定的网络压缩文件，则不会解压。
- **VOLUME:**
在镜像里创建一个指定路径(文件或文件夹)的挂载点，这个容器可以来自主机或者其它容器。多个容器可以通过同一个挂载点共享数据，即便其中一个容器已经停止，挂载点也仍可以访问。
- **WORKDIR:**
为接下来执行的指令指定一个新的工作目录，这个目录可以是绝对目录，也可以是相对目录。根据需要，WORKDIR可以被多次指定。当启动一个容器时，最后一条WORKDIR指令所指的目录将作为容器运行的当前工作目录。
- **ENV:**
设置容器运行的环境变量。在运行容器的时候，通过设置-e参数可以修改这个环境变量值，也可以添加新的环境变量。
例如：
docker run -e WEBAPP_PORT=8000 -e WEBAPP_HOST=www.example.com ...
- **CMD:**
用来设置启动容器时默认运行的命令。

- **ENTRYPOINT:**
用来指定容器启动时的默认运行的命令。区别在于：运行容器时添加在镜像之后的参数，对ENTRYPOINT是拼接，CMD是覆盖。
 - 若在Dockerfile中指定了容器启动时的默认运行命令为**ls -l**，则运行容器时默认启动命令为**ls -l**，例如：
 - **ENTRYPOINT ["ls", "-l"]:** 指定容器启动时的程序及参数为**ls -l**。
 - **docker run centos:** 当运行centos容器时，默认执行的命令是**docker run centos ls -l**
 - **docker run centos -a:** 当运行centos容器时拼接了-a参数，则默认运行的命令是**docker run centos ls -l -a**
 - 若在Dockerfile中指定了指定了容器启动时的默认运行命令为**--entrypoint**，则在运行容器时若需要替换默认运行命令，可以通过添加**--entrypoint**参数来替换Dockerfile中的指定。例如：
docker run gutianlangyu/test --entrypoint echo "hello world"
- **USER:**
为容器的运行及RUN、CMD、ENTRYPOINT等指令的运行指定用户或UID。
- **ONBUILD:**
触发器指令。构建镜像时，容器引擎的镜像构建器会将所有的ONBUILD指令指定的命令保存到镜像的元数据中，这些命令在当前镜像的构建过程中并不会执行。只有新的镜像使用FROM指令指定父镜像为当前镜像时，才会触发执行。
使用FROM以这个Dockerfile构建出的镜像为父镜像，构建子镜像时：
ONBUILD ADD . /app/src: 自动执行**ADD . /app/src**

9.1.3 如何制作镜像压缩包？

使用docker save命令可将容器镜像制作成tar或tar.gz文件压缩包，具体命令格式如下：

```
docker save [OPTIONS] IMAGE [IMAGE...]
```

OPTIONS说明：--output、-o，表示导出到文件。

示例：

```
$ docker save nginx:latest > nginx.tar
$ ls -sh nginx.tar
108M nginx.tar

$ docker save php:5-apache > php.tar.gz
$ ls -sh php.tar.gz
372M php.tar.gz

$ docker save --output nginx.tar nginx
$ ls -sh nginx.tar
108M nginx.tar

$ docker save -o nginx-all.tar nginx
$ docker save -o nginx-latest.tar nginx:latest
```

9.2 镜像管理类

9.2.1 长期有效的登录指令与临时登录指令的区别是什么？

- 临时的登录指令代指24个小时后会过期失效，不能再被使用的登录指令。
- 长期有效的登录指令有效期为永久。

获取了长期有效的登录指令后，在有效期内的临时登录指令仍然可以使用。

长期有效的登录指令与临时登录指令均不受限制，可以多台机器多人同时登录。

9.2.2 为什么通过客户端和页面上传的镜像大小不一样？

问题描述

假设在本地Docker客户端上制作了一个nginx镜像，版本号为v2.0.0，使用**docker images**命令查询SIZE为**22.8MB**：

```
$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
nginx                v2.0.0      22f2bf2e2b4f 9 days ago   22.8MB
```

1. 通过客户端（即执行**docker push**命令）上传该镜像至SWR镜像仓库，查询大小为**9.5MB**。

| 镜像版本 | 大小 | 更新时间  |
|--------|--------|--|
| v2.0.0 | 9.5 MB | 2021/09/01 19:46:12 GMT+08:00 |

2. 在本地Docker客户端将镜像打包为tar格式，将nginx.tar下载至本地后，通过页面方式上传至SWR镜像仓库，查询大小为**23.2MB**。

| 镜像版本 | 大小 | 更新时间  |
|--------|---------|--|
| v2.0.0 | 23.2 MB | 2021/09/10 09:59:28 GMT+08:00 |

可以发现，通过客户端和页面上传的镜像大小不一样。

原因分析

使用客户端上传的镜像每一层layer都进行了tgz压缩，而页面上传的镜像包是经过Docker打包的，每一层layer只进行了打包，没有压缩。所以两种方式上传的镜像大小显示会不一致。

9.2.3 控制台页面的容器镜像可以下载到本地吗？

控制台页面的容器镜像不能直接下载至本地，您可以参考以下方法操作：

1. 在镜像详情页获取镜像的下载指令。
2. 在安装了Docker客户端的机器上执行上一步的指令下载镜像。

示例：

```
docker pull swr.ae-ad-1.myhuaweicloud.com/group/nginx:v1
```

3. 将镜像保存为.tar或.tar.gz格式的文件。

示例：

```
docker save nginx:v1 > nginx.tar
```

4. 下载文件至本地。

9.3 故障类

9.3.1 为什么登录指令执行失败？

登录指令执行失败有以下几种情况：

1. 容器引擎未安装正确，报如下图所示错误：

“docker: command not found”

解决方法：重新安装容器引擎。

- 由于容器镜像服务支持容器引擎1.11.2及以上版本上传镜像，建议下载对应版本。
- 安装容器引擎需要连接互联网，内网服务器需要绑定弹性IP后才能访问。

2. 临时登录指令已过期或登录指令中区域项目名称、AK、登录密钥错误，报如下图所示错误：

“unauthorized: authentication required”

解决方法：登录容器镜像服务控制台，在左侧菜单栏选择“我的镜像”，单击右侧“客户端上传”获取登录指令。

- a. 获取临时的登录指令：单击“生成临时登录指令”，在弹出的页面中单击复制登录指令。
- b. 获取长期有效的登录指令：单击“如何获取长期有效登录指令”，参考其中的指导获取。

3. 登录指令中镜像仓库地址错误，报如下所示错误：

“Error logging in to v2 endpoint, trying next endpoint: Get https://{{endpoint}}/v2/: dial tcp: lookup {{endpoint}} on xxx.xxx.xxx.xxx:53 : no such host”

解决方法：

- a. 修改登录指令中的镜像仓库地址。
- b. 获取临时的登录指令：方法请参见2。

4. **x509: certificate has expired or is not yet valid**

长期有效登录指令中AK/SK被删除导致，请使用有效的AK/SK生成登录指令。

5. **x509: certificate signed by unknown authority**

问题原因：

容器引擎客户端和SWR之间使用HTTPS的方式进行通信，客户端会对服务端的证书进行校验。如果服务端证书不是权威机构颁发的，则会报如下错误：x509: certificate signed by unknown authority

```
[root@luster01-fd0rb ~]# docker login -u cn-north-1@SWR-MINTIAIXUWBUGPF -p 5cad07aa2ba0b2956c29f5fe1173abe909f226fa509f7ab14dc85d749569d2 registry.cn-north-1.huaweiclouds.com
Error response from daemon: Get https://registry.cn-north-1.huaweiclouds.com/v1/users/: x509: certificate signed by unknown authority
```

解决方法：

如果用户信赖服务端，跳过证书认证，那么可以手动配置Docker的启动参数，配置方法如下：

- CentOS:

修改“/etc/docker/daemon.json”文件（如果没有，可以手动创建），在该文件内添加如下内容：

```
{  
  "insecure-registries": [{"镜像仓库地址"}]  
}
```

- Ubuntu:

修改“/etc/default/docker”文件，在DOCKER_OPTS配置项中增加如下内容：

```
DOCKER_OPTS="--insecure-registry {镜像仓库地址}"
```

- EulerOS:

修改“/etc/sysconfig/docker”文件，在INSECURE_REGISTRY配置项中增加如下内容：

```
INSECURE_REGISTRY='--insecure-registry {镜像仓库地址}'
```

📖 说明

镜像仓库地址支持域名和IP形式。

- 域名形式的镜像仓库地址获取方式：参考[2](#)获取临时登录指令，末尾的域名即为镜像仓库地址。
- IP：可通过ping镜像仓库地址（域名形式）获取。

配置完成后，执行**systemctl restart docker**重启容器引擎。

9.3.2 为什么使用客户端上传镜像失败？

denied: you do not have the permission

问题现象：使用客户端上传镜像，报如下所示错误：

“denied: you do not have the permission”

问题原因：

- 该组织名已被其他用户注册或当前SWR组织数量已超过配额。
- docker login命令使用IAM用户的AK、SK生成，没有对应组织的权限。

解决方法：

- 该组织名已被其他用户注册时：建议您先创建组织然后再上传镜像。
- SWR组织数量超过配额时：单个用户的组织数量限制为5个，您可以将镜像上传到已存在的组织下。
- 没有对应组织的权限：使用帐号授权后，可以正常推送。

tag does not exist: xxxxxx 或 An image does not exist locally with the tag: xxxxxx

问题现象：使用客户端上传镜像，报如下图所示错误：

“tag does not exist: xxxxxx”

或

“An image does not exist locally with the tag: xxxxxx”

问题原因：上传的镜像或镜像版本不存在。

解决方法：通过docker images查看本地镜像，确认要上传的镜像名称及版本后，重新上传镜像。

name invalid: 'repository' is invalid

问题现象：使用客户端上传镜像，报如下图所示错误：

“name invalid: 'repository' is invalid”

问题原因：组织命名或镜像命名不规范。

解决方法：以下分别是组织名（namespace）和仓库名（repository）的命名正则表达式：

namespace: $^([a-z]+(?:_(?:[a-z0-9]+)?)?)$$ ，长度范围为：1-64；

repository: $^([a-z0-9]+(?:_(?:[a-z0-9]+)?)?)$$ ，长度范围为：1-128。

您可以按照上述命名规范，重新指定上传的组织 and 镜像名称。

偶现推送镜像超时

问题现象：偶现推送镜像超时

问题原因：您在国内机器往国外推镜像，网速慢，偶现连接失败。

9.3.3 为什么通过页面上传镜像失败？

SWR对镜像的命名和地址有严格的规范。如果镜像的命名不规范或镜像地址不规范都会导致镜像上传失败。

镜像格式不合法

问题现象：通过页面上传镜像，出现“镜像格式不合法”的报错。

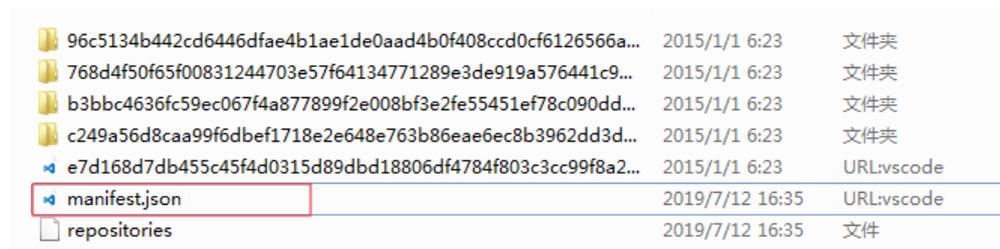
问题原因：镜像地址不规范，导致上传失败。

镜像地址各个部分的含义如下，最后的tag（版本号）可省略，如果省略则表示latest版本，其余部分均不可省略，且不可多余。

样例：registry.example.com/repo_namespace/repo_name:tag

- registry.example.com为容器镜像服务的镜像仓库地址，此处为示例，以实际地址为准。
- repo_namespace为组织名称，命名正则表达式为 $^([a-z]+(?:_(?:[a-z0-9]+)?)?)$$ ，长度范围为：1-64。
- repo_name:tag为镜像名称和版本号，镜像命名正则表达式为 $^([a-z0-9]+(?:_(?:[a-z0-9]+)?)?)$$ ，长度范围为：1-128。

您可以将镜像解压，打开文件manifest.json文件查看RepoTags字段的值是否符合上述规范。



| | | |
|---|-----------------|-----------|
| 96c5134b442cd6446dfae4b1ae1de0aad4b0f408ccd0cf6126566a... | 2015/1/1 6:23 | 文件夹 |
| 768d4f50f65f00831244703e57f64134771289e3de919a576441c9... | 2015/1/1 6:23 | 文件夹 |
| b3bbcc4636fc59ec067f4a877899f2e008bf3e2fe55451ef78c090dd... | 2015/1/1 6:23 | 文件夹 |
| c249a56d8caa99f6dbef1718e2e648e763b86eae6ec8b3962dd3d... | 2015/1/1 6:23 | 文件夹 |
| e7d168d7db455c45f4d0315d89dbd18806df4784f803c3cc99f8a2... | 2015/1/1 6:23 | URLvscode |
| manifest.json | 2019/7/12 16:35 | URLvscode |
| repositories | 2019/7/12 16:35 | 文件 |

解决方法：按照命名规范，重新给镜像打tag，然后使用docker save命令保存镜像，然后再使用页面上传。

一直卡在上传界面直到超时

问题现象：通过页面上传镜像，一直卡在上传界面直到超时。

问题原因：镜像命名不规范，导致上传失败。

解决方法：您可以按照镜像命名规范修改镜像名称后，重新上传镜像。

须知

SWR判定镜像名是否合法不是以用户在界面上传镜像时的文件名为依据，而是依据镜像包中的repositories和manifest.json文件。

9.3.4 为什么 docker pull 指令执行失败？

x509: certificate signed by unknown authority

问题现象：使用docker pull拉取镜像，报错“x509: certificate signed by unknown authority”。

问题原因：

- 容器引擎客户端和SWR之间使用HTTPS的方式进行通信，client会对服务端的证书进行校验，如果客户端安装的根证书不完整，会报如下错误：“x509: certificate signed by unknown authority”。
- 容器引擎客户端配置了Proxy导致。

解决方法：

- 如果您信赖服务端，跳过证书认证，那么可以手动配置容器引擎的启动参数，配置如下（其中地址配置成需要的即可，选择一个配置即可）：
 - /etc/docker/daemon.json（如果没有可以手动创建），在该文件内添加如下配置（注意缩进，2个空格）：

```
{
  "insecure-registries":["镜像仓库地址"]
}
```
 - /etc/sysconfig/docker：

```
INSECURE_REGISTRY='--insecure-registry=镜像仓库地址'
```添加配置后执行如下命令重启：**systemctl restart docker**或**service restart docker**。
- 执行**docker info**命令，检查proxy配置是否正确，修改为正确的proxy配置。

Error: remote trust data does not exist

问题现象：使用docker pull拉取镜像，报错“Error: remote trust data does not exist”。

问题原因：客户端开启镜像签名验证，而镜像没有镜像签名层。

解决方法： 查看环境变量是否设置了**DOCKER_CONTENT_TRUST=1**，如果设置了，请修改“/etc/profile”文件将**DOCKER_CONTENT_TRUST=1**去掉，然后执行**source /etc/profile**生效。

A 修订记录

| 发布日期 | 更新特性 |
|------------|----------|
| 2020-11-05 | 第一次正式发布。 |