

容器镜像服务

用户指南 (安卡拉区域)

文档版本 01
发布日期 2024-12-02



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。

安全声明

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>

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 权限管理	7
3.1 创建用户并授权使用 SWR	7
4 容器引擎基础知识	9
5 镜像管理	12
5.1 客户端上传镜像	12
5.2 获取 docker 容器引擎长期有效登录指令	14
5.3 获取 containerd 容器引擎长期有效的拉取、推送镜像指令	15
5.4 页面上传镜像	16
5.5 下载镜像	17
5.6 编辑镜像属性	18
5.7 共享私有镜像	19
5.8 添加触发器	20
5.9 添加镜像老化规则	22
6 组织管理	24
7 授权管理	26
8 常见问题	29
8.1 通用类	29
8.1.1 什么是容器镜像服务?	29
8.1.2 产品咨询	29
8.1.3 如何制作容器镜像?	30
8.1.4 如何制作镜像包?	34
8.1.5 SWR 的配额是多少?	34

8.1.6 为什么创建组织失败?	34
8.2 登录问题.....	35
8.2.1 长期有效的登录指令与临时登录指令的区别是什么?	35
8.2.2 登录指令提示过期该怎么办?	35
8.3 镜像上传.....	35
8.4 镜像下载.....	36
8.5 镜像管理类.....	37
8.5.1 容器镜像服务是否可以拉取容器镜像到本地?	37
8.6 故障类.....	37
8.6.1 为什么登录指令执行失败?	37
8.6.2 为什么使用客户端上传镜像失败?	39
8.6.3 为什么通过页面上传镜像失败?	40
8.6.4 为什么 docker pull 指令执行失败?	41
8.6.5 API 调用异常.....	42
8.7 其他.....	42
8.7.1 为什么 CCE 工作负载拉取 SWR 内的镜像异常, 且提示为“未登录”?	42
8.7.2 为什么通过客户端和页面上传的镜像大小不一样?	43
8.7.3 SWR 私有镜像最多可以共享给多少个租户?	43

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。

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，提供账号间、账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。

1.3 应用场景

镜像生命周期管理

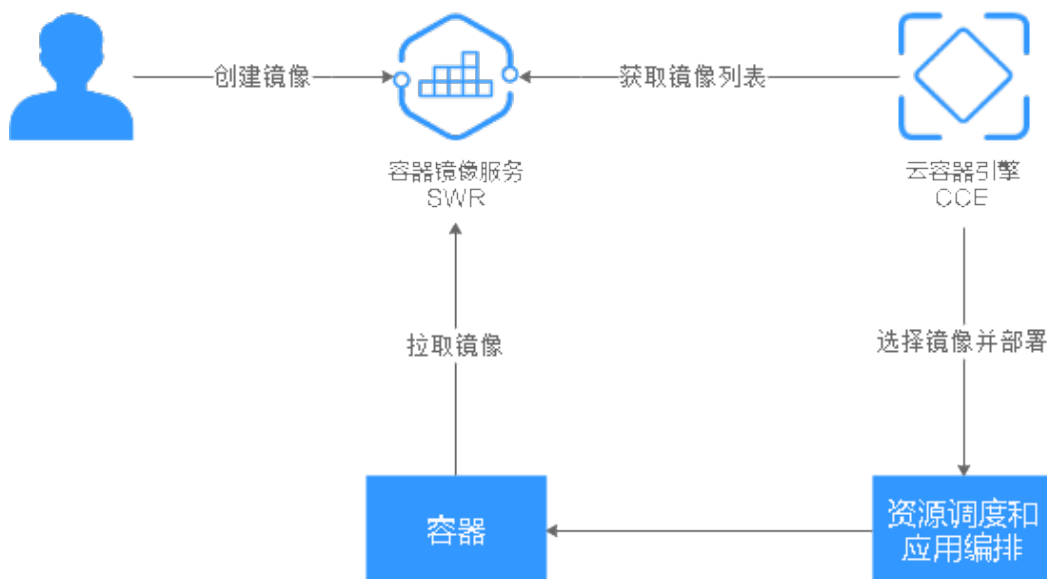
提供镜像构建、镜像上传、下载、同步、删除等完整的生命周期管理能力。

优势

- 加速下载：利用镜像下载加速技术，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。

建议搭配云容器引擎CCE使用

图 1-1 SWR 与 CCE 配合使用示意图



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。支持为账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所能创建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，单个租户同时上传镜像layer总数不大于20个。
- 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。

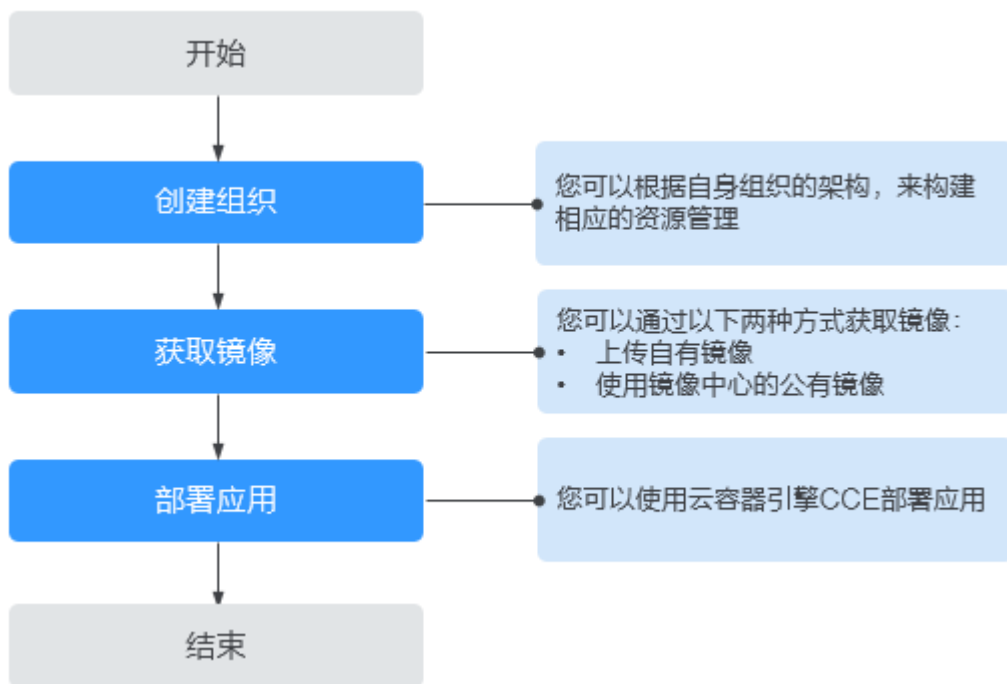
- 云容器引擎
云容器引擎 (Cloud Container Engine, 简称CCE) 提供高可靠高性能的企业级容器应用管理服务, 支持Kubernetes社区原生应用和工具, 简化云上自动化容器运行环境搭建。
容器镜像服务能无缝对接CCE, 您可以将容器镜像服务中的镜像部署到CCE中。

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 权限管理

3.1 创建用户并授权使用 SWR

如果您需要对您所拥有的容器镜像服务（SWR）进行精细的权限管理，您可以使用统一身份认证服务（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用SWR资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将SWR资源委托给更专业、高效的其他云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用SWR服务的其他功能。

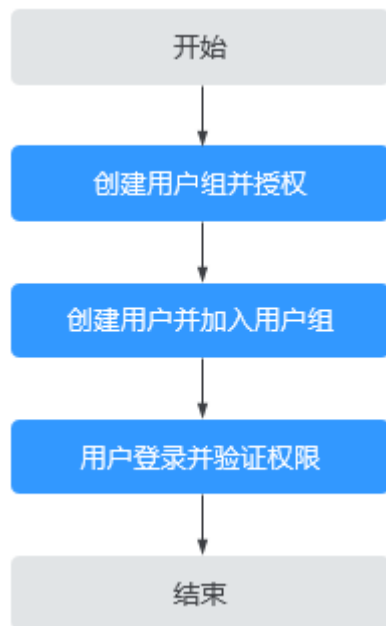
本章节为您介绍对用户授权的方法，操作流程如[图3-1](#)所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的SWR权限，并结合实际需求进行选择。

示例流程

图 3-1 给用户授予 SWR 权限流程



1. 创建用户组并授权。
在IAM控制台创建用户组，并在“操作”列下选择“授权”，并授予容器镜像服务的管理员权限“SWR Administrator”。
2. 创建用户并加入用户组。
在IAM控制台创建用户，并在“操作”列下选择“授权”，将其加入1中创建的用户组。
3. 用户登录并验证权限。
新创建的用户登录控制台，切换至授权区域，验证权限（如果能顺利完成如下操作，说明权限设置成功）：
 - a. 在“服务列表”中选择容器镜像服务，进入SWR主界面。
 - b. 在左侧导航栏选择“组织管理”，单击右上角“创建组织”，输入组织名称，能够成功创建组织。
 - c. 在左侧导航栏选择“我的镜像”，单击右上角“页面上传”，选择上一步创建的组织，以及一个本地的镜像文件，能够成功上传镜像。

4 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容器引擎 1.11.2（包含）到24.0.9（包含）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。

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

导入镜像文件

本章节将指导你通过docker load命令将镜像压缩包导入为一个镜像。

执行方式有2种:

docker load < 路径/文件名.tar

docker load --input或者-i 路径/文件名.tar

示例:

```
$ docker load --input fedora.tar
```


5 镜像管理

5.1 客户端上传镜像

操作场景

客户端上传镜像，是指在安装了容器引擎客户端的机器上使用docker命令或者ctr命令将镜像上传到容器镜像服务的镜像仓库。如果是docker容器引擎客户端则使用docker push命令上传。如果是containerd容器引擎客户端则使用ctr push命令上传。

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。
- 若使用docker容器引擎客户端上传镜像，请确保docker容器引擎客户端版本必须为1.11.2（包含）到24.0.9（包含）之间的。


前提条件

已创建组织，请参见[创建组织](#)。

docker 容器引擎客户端

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。操作步骤如下：

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docker容器引擎长期有效登录指令](#)。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 {Image repository address}/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 {Image repository address}/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

containerd 容器引擎客户端

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。

步骤3 在镜像详情页面中，进入“Pull/Push指南”页签，复制containerd容器引擎的镜像上传指令。

📖 说明

该指令将于6个小时后过期。如果想获取长期上传指令请参考[获取containerd容器引擎长期有效的拉取、推送镜像指令](#)。

步骤4 以root用户登录containerd引擎所在的虚拟机。

步骤5 在虚拟机中执行3复制的镜像上传指令，请修改版本名称为要上传镜像的。

```
[root@ ~]# docker image push --user sa-fb-1_8582V7B5UJ2955AM7MB9Y:5b73d8e8d7f267bacb3e94fbb1c7a498f5ffc2ce1538856045f41dc37beb4_sw_sa-fb-1_rguian2out.com/test/test:v2
WARN[000] DEPRECATION: The 'mirrors' property of '[plugins]-id.containerd.grpc.v1.cri--registries' is deprecated since containerd v1.5 and will be removed in containerd v2.0. Use 'config_path' instead.
WARN[000] DEPRECATION: The 'config' property of '[plugins]-io.containerd.grpc.v1.cri--registries' is deprecated since containerd v1.5 and will be removed in containerd v2.0. Use 'config_path' instead.
mani fest-sha256:5e9f5a31de11138255a7db015d6121458963e8fa0c4d99c83e3754f9dd07: done
config-sha256:150bdf6d3701a212c8ab9dac676a21ff7c7a08380dd585cfd08528d64ef8e: done
total: 1.7 KiB (5.5 KiB/s)
Upload: 0.3 s
```

步骤6 检查镜像是否上传成功。

----结束

5.2 获取 docker 容器引擎长期有效登录指令

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。

操作流程

您可以按照以下流程获取长期有效登录指令。

图 5-1 流程示意图



操作步骤

步骤1 获取编程访问权限。(如果当前用户已有编程访问权限，请忽略此步骤)

1. 以管理员身份，登录管理控制台。
2. 在管理控制台左上角单击📍，选择区域和项目。
3. 单击左侧导航栏☰，选择“管理与部署” > “统一身份认证服务IAM”。
4. 在“用户”页搜索框输入并搜索要授予编程访问权限的用户名称。
5. 单击用户名称，进入用户详情页。
6. 单击“访问方式”后面的✎按钮。
7. 勾选“编程访问”选项。（可单独勾选编程访问，也可以2种访问方式同时勾选。）

步骤2 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“API凭证”的项目列表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3 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。

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. 在左侧导航栏中选择“访问密钥”，单击“新增访问密钥”。
3. 单击“确定”，下载访问密钥，其中包含AK和SK。

📖 说明

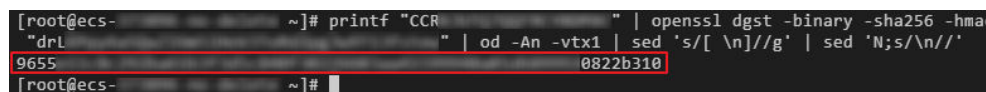
为防止访问密钥泄露，建议您将其保存到安全的位置。

步骤4 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3**获取的AK/SK。

图 5-2 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5 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2**中获取，AK在**步骤3**中获取，登录密钥为**步骤4**的执行结果。

📖 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6 使用history -c命令清理相关使用痕迹，避免隐私信息泄露。

----结束

5.3 获取 containerd 容器引擎长期有效的拉取、推送镜像指令

操作场景

本章节介绍如何获取containerd容器引擎长期有效的拉取、推送镜像指令，长期有效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。
- 用户登录IAM控制台前，请确保已具有IAM服务访问权限。

操作步骤

步骤1 参考**获取docker容器引擎的长期有效登录指令**中的**步骤1**设置编程访问权限。

步骤2 参考**获取docker容器引擎的长期有效登录指令**中的**步骤2至步骤4**获取资源空间名、镜像仓库地址、AK以及登录密钥信息。

步骤3 使用如下的格式拼接长期拉取和推送镜像的指令。

1. 镜像拉取指令拼接

```
ctr image pull --user [资源空间名]@[AK]:[登录密钥] [镜像仓库地址]
```

其中，资源空间名和镜像仓库地址在**步骤2**中获取，AK在**步骤3**中获取，登录密钥为**步骤4**的执行结果。

2. 镜像推送指令拼接

```
ctr image push --user [资源空间名]@[AK]:[登录密钥] [镜像仓库地址]
```

其中，资源空间名和镜像仓库地址在**步骤2**中获取，AK在**步骤3**中获取，登录密钥为**步骤4**的执行结果。

说明

- 登录密钥字符串是经过加密的，无法将登录密钥字符串逆向解密成SK。
- 获取的指令可在其他机器上使用并进行镜像上传下载。

----结束

5.4 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（包含）到24.0.9（包含）版本Docker容器引擎客户端制作的镜像压缩包。

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。

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

5.5 下载镜像

操作场景

您可以使用docker容器引擎也可以使用containerd容器引擎下载容器镜像服务中的镜像。


docker 容器引擎

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。

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

containerd 容器引擎

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。

步骤3 在镜像详情页面中，复制操作列的“containerd指令”或者进入“Pull/Push指南”页签，复制containerd容器引擎的镜像下载指令。

说明

该指令将于6个小时后过期。如果想获取长期下载指令请参考[获取containerd容器引擎长期有效的拉取、推送镜像指令](#)。

步骤4 以root用户登录containerd引擎所在的虚拟机。

步骤5 在虚拟机中执行**3**复制的镜像下载指令。

- 复制操作列的“containerd指令”的场景下执行如下：

```
[root@ ~]# crictl pull --creds
9be186eac1d48ea2fbfb68f15282_svr..._mjhuacloud.com/.../kritis-validation-admission:24.3.14_aarch64
Image is up to date for sha256:68422ad99915e044af2a15ef0948c958f98844c220924d26726e9ba63287b6cb
```

- 复制“Pull/Push指南”页签containerd容器下载指令的场景下请修改版本名称为要上传镜像的再执行。

```
[root@ ~]# ctr image pull --user
518156cca1cf9fa9b0f6b24e748f6c swr.../mjuaweicloud.com/~/kritis-validation-admission:24.3.14_aarch64
WARN[0000] DEPRECATION: The 'mirrors' property of '[plugins."io.containerd.grpc.v1.cri".registry]' is deprecated since container
d v1.5 and will be removed in containerd v2.0. Use 'config_path' instead.
swr.cn-north-7.mjuaweicloud.com/~/kritis-validation-admission:24.3.14_aarch64: resolved |.....|
|.....|
manifest-sha256:e34f8144b428fdaff2a9f82f2aa9d5291a04340a408ea166da9f9cee91be2d32: done |.....|
|.....|
config-sha256:68d22ad99915e044af2a15ef8948c958f9884dc228924d26726e9ba63287b6cb: done |.....|
|.....|
layer-sha256:4ec0f98f626c28abeab80113f5994a423bc20b7e9b107e85a0c0f2a3d40e47be: done |.....|
|.....|
layer-sha256:bc7f43b68355f37558f4e315f2b1bd64d03412622cd66e794036db13324fb7ce: done |.....|
|.....|
layer-sha256:825d167cd3557957e6c3ef139683e897102ddeefbd7c2886359e2f54e88cbd7e: done |.....|
|.....|
layer-sha256:3f954aa41fd454743e8d894744d03ec38635f7387410bb498a1d2b63b6fe74ab: done |.....|
|.....|
layer-sha256:51482c154beec8d31a697f886740de512aeecc022c0d4a88fe838d31a3782: done |.....|
|.....|
elapsed: 6.1 s total: 372.7 (61.1 MiB/s)
unpacking linux/arm64 sha256:e34f8144b428fdaff2a9f82f2aa9d5291a04340a408ea166da9f9cee91be2d32...
done: 1.293649364s
```

步骤6 查看镜像是否下载成功。

- 复制操作列的“containerd指令”的场景下使用crictl images 命令查看是否下载成功。

```
[root@ ~]# crictl pull --creds c...
9be186eac1d40ea2fb668f15282 swr.../mjuaweicloud.com/~/kritis-validation-admission:24.3.14_aarch64
Image is up to date for sha256:68d22ad99915e044af2a15ef8948c958f9884dc228924d26726e9ba63287b6cb
[root@ ~]# crictl images
IMAGE                                TAG                                IMAGE ID                                SIZE
docker.io/library/cce-pause          3.1                                c96888c71666e                          687KB
swr.cn-north-7.mjuaweicloud.com/~/kritis-validation-admission 24.3.14_aarch64 68d22ad99915e                          394MB
swr.cn-north-7.mjuaweicloud.com/huofficial/everest-csi-driver-init 2.4.61          f1e3147427fcf                          129MB
swr.cn-north-7.mjuaweicloud.com/huofficial/everest                2.4.61          24f984194b3af                          186MB
swr.cn-north-7.mjuaweicloud.com/op_svc_apm/icagent               5.31.32.41     b9f53a98a0d1b                          219MB
```

- 复制“Pull/Push指南”页签containerd容器下载指令的场景下使用ctr images list命令查看是否下载成功。

```
[root@ ~]# ctr images list
WARN[0000] DEPRECATION: The 'mirrors' property of '[plugins."io.containerd.grpc.v1.cri".registry]' is deprecated since container
d v1.5 and will be removed in containerd v2.0. Use 'config_path' instead.
REF                                TYPE                                SIZE                                PLATFORMS                                LABELS
swr.cn-north-7.mjuaweicloud.com/h30013402/kritis-validation-admission:24.3.14_aarch64 application/vnd.docker.distribution.manif
est.v2+json sha256:e34f8144b428fdaff2a9f82f2aa9d5291a04340a408ea166da9f9cee91be2d32 375.5 MiB linux/arm64 -
```

----结束

5.6 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”）、类别及描述。

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”）、类别及描述，然后单击“确定”。

表 5-1 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。
类别	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

5.7 共享私有镜像

操作场景

镜像上传后，您可以共享**私有镜像**给其他账号，并授予下载该镜像的权限。

被共享的账号下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，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5-2填写相关参数，然后单击“确定”。

表 5-2 共享镜像

参数	说明
共享给	输入账号名称。
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，选择“我共享的镜像”，查看所有共享的镜像。

----结束

5.8 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5-3填写相关参数，然后单击“确定”。

表 5-3 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。

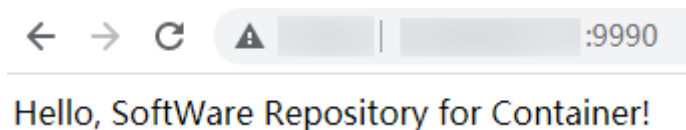
图 5-3 nginx 无状态负载



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v3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!”。

图 5-4 更新后的 nginx



5.9 添加镜像老化规则

操作场景

镜像上传后，您可以添加镜像老化规则。容器镜像服务提供了如下两种类型的镜像老化处理规则，规则设置完成后，系统会根据已定义的规则自动执行镜像老化操作。

- 存活时间：设置该类型的老化规则后，留存时间超过指定时间的老旧镜像将被删除。
- 版本数目：设置该类型的老化规则后，留存镜像超过指定值时，老旧镜像将被删除。

此外，对于特定版本的镜像可通过添加过滤策略来保留，免受老化规则的影响。

约束与限制

一个镜像仅支持添加一个老化规则。如需添加新的老化规则，需要删除已有老化规则。

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“镜像老化”页签，单击“+”，根据表5-4填写相关参数，然后单击“确定”。

表 5-4 添加镜像老化规则

参数	说明
规则类型	分为存活时间和版本数目。 <ul style="list-style-type: none">• 存活时间：设置该类型的老化规则后，留存时间超过指定时间的老旧镜像将被删除。• 版本数目：设置该类型的老化规则后，留存镜像超过指定值时，老旧镜像将被删除。
保留天数	镜像留存的最大天数，可设置为1~365的整数。规则类型设置为“存活时间”时，需要配置此参数。
保留数目	镜像留存的最大数目，可设置为1~1000的整数。规则类型设置为“版本数目”时，需要配置此参数。

参数	说明
过滤标签	输入将被过滤的镜像版本，在应用老化规则前指定版本的镜像将被过滤掉。
过滤正则	输入将被过滤的版本正则式，在应用老化规则前所有版本号满足正则表达式的镜像将被过滤掉。

镜像老化规则添加成功后，系统会立即进行一次查询，清理掉符合老化规则的镜像，且在“老化日志”中显示清理结果。

----结束

6 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6-1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 6-1 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”。

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。

----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，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。

7 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 Administrator或者Tenant Administrator系统权限时，您就拥有了SWR的账号管理员权限，可以在SWR中为其他IAM用户进行授权。

如果您没有SWR的账号管理员权限，就需要已拥有SWR账号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。

说明

- 拥有SWR账号管理员权限的用户，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。

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 7-1 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。

----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”，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”，然后单击“确定”。

8 常见问题

8.1 通用类

8.1.1 什么是容器镜像服务？

容器镜像服务（SoftWare Repository for Container，简称SWR）是一种支持容器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，帮助用户快速部署容器化服务。

8.1.2 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。

容器镜像服务的带宽多大？

容器镜像服务的带宽会根据用户使用情况动态变化。

请问 SWR 是否支持查询一个镜像的 cpu 架构（x86 or ARM）？

- 对于公共镜像，您可以登录SWR控制台，进入镜像中心，搜索资源并查看镜像的详细信息，包括该镜像支持的架构。
- 对于您的私有镜像，您可以使用`docker inspect` [镜像名称：版本名称]，查询该镜像的架构。

示例：`docker inspect openjdk:7`。

图 8-1 示例

```
},  
  "Architecture": "amd64",  
  "Os": "linux",  
  "Size": 621209007,  
  "VirtualSize": 621209007,  
  "GraphDriver": {  
    "Data": {  
      "LowerDir": "/var/lib/docker/overlay2/93ff8e16f44919ca8ec9e5c5077ea166c79a3f5f1dbf46288390a77d3cedf7b/diff:/var  
/lib/docker/overlay2/579337e171a09f26649010fadac542b9b050079fda9a97837450c20ebc36076e/diff:/var/lib/docker/overlay2/d9b3e45a1339  
74fc00a8a78c4462f066b56ba3d3af23fa4ba59cc1cf6efafb2/diff",  
      "MergedDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7408e1b76852/merged",  
      "UpperDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7408e1b76852/diff",  
      "WorkDir": "/var/lib/docker/overlay2/09ead26ee61e1ecfd8556963d768e888a63b8f51dc506b4c806c7408e1b76852/work"  
    },  
    "Name": "overlay2"  
  },  
  "RootFS": {  
    "Type": "layers",  
    "Layers": [  
      "sha256:174f5685490326fc8a1c8f5570b8663732189b327007e47ff13d2ca59673db02",  
      "sha256:fd431e3bcfb54e46ee912e04becffa279cd866c278ef74dd0a15fca05dcd9b",  
      "sha256:f0200dd2ca8ff94a8beacb8df885ff2c9c4843dac363abea052181063b535445",  
      "sha256:0f137c758756a06bc5bd64d05ceb636525f1267fbef49b8147f651a062b54ea7"  
    ]  
  }  
}
```

8.1.3 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。

图 8-2 流程示意图



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。

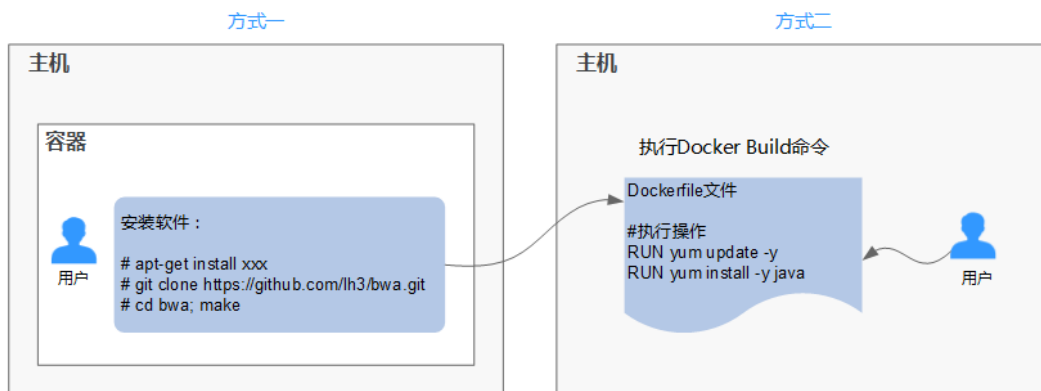
6. 执行docker images可以查看到制作完成的容器镜像。

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。

图 8-3 使用 Dockerfile 方式构建



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用户为后续命令的執行者
USER root

#執行操作
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，也可以使一个文件夹。
 - 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**

8.1.4 如何制作镜像包？

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
```

8.1.5 SWR 的配额是多少？

容器镜像服务对镜像数量没有配额限制，您可以根据需要上传镜像。

容器镜像服务对单个用户的组织数量限定了配额，如表8-1所示。

表 8-1 容器镜像服务配额

资源类型	配额
组织	5

8.1.6 为什么创建组织失败？

问题现象：创建组织失败，页面提示该组织已经存在，但在“组织管理”页面没有查询到该组织。

解决办法：组织名称全局唯一，即当前区域下，组织名称唯一。

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。

8.2 登录问题

8.2.1 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6个小时后会过期失效，不能再被使用的登录指令。可应用在临时使用，对外单次授权等场景中，对安全性要求较高的生产集群也可通过定时刷新的方式进行使用。
- 长期有效的登录指令有效期为永久。可应用在前期测试、CICD流水线及容器集群拉取镜像等场景中。

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。

8.2.2 登录指令提示过期该怎么办？

清除浏览器缓存后重新生成登录指令即可，临时登录指令会在6个小时后失效。

8.3 镜像上传

如何通过 API 上传镜像到 SWR？

SWR暂时没有开放镜像上传的API，您可以使用docker push或页面方式上传镜像。

为什么通过客户端上传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**。
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**。

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。

SWR 镜像仓库支持 ARM 镜像上传么？

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。

SWR 单个 layer 的最大限制是多少?

使用客户端向SWR上传镜像，镜像的每个layer大小不能超过10G。

SWR 公网上传单租户流控限制是多少?

为了确保SWR镜像上传租户间不会互相影响，当前对单个租户流控为镜像层上传20QPS，超过后会被限流，docker会收到503并会自动重试限流拦截的请求。

SWR 支持断点续传镜像文件吗?

暂不支持。

8.4 镜像下载

如何通过 API 下载镜像?

SWR暂不支持通过API下载镜像。如果要下载镜像，请通过docker pull命令来进行下载操作。

docker pull 下载的镜像存放在什么地方?

docker pull将镜像下载到本地节点上，您可以通过docker save命令将镜像保存成tar归档文件。

是否支持跨区域下载镜像?

支持。

SWR当前支持跨区域公网下载，请确保获取正确的登录指令。

控制台页面的镜像可以下载到本地吗?

控制台页面的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。
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。

拉取镜像慢可能的原因

1. 网络原因。
2. 镜像可能是多层的。
3. 拉取镜像任务可能是串行的，前面的拉取镜像任务未完成的情况下后面的拉取任务将会等待。前面的拉取任务的超时时间是30分钟。

同名同 tag 的镜像重复拉取会覆盖原有旧的镜像吗？

如果镜像manifest未发生变化，不会覆盖；如果镜像manifest发生变化，会覆盖。

8.5 镜像管理类

8.5.1 容器镜像服务是否可以拉取容器镜像到本地？

存储在SWR中的容器镜像不支持通过控制台直接下载。拉取镜像的操作步骤如下：

1. 在镜像详情页获取镜像拉取命令。
2. 在安装Docker客户端的设备上执行拉取的命令。

示例：

```
docker pull {镜像仓库地址}/group/nginx:v1
```

3. 将图像另存为TAR或TAR.GZ文件。

示例：

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

4. 将文件下载到本地。

8.6 故障类

8.6.1 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Docker容器引擎1.11.2（包含）到24.0.9（包含）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

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。

6. **denied: Not allow to login、upload or download image**

用户大批量并发上传镜像或者攻击服务，系统把用户拉黑，用户无法登录和上传下载镜像。请在30分钟之后重新尝试。

8.6.2 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: 使用账号授权后, 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。

不允许用户上传镜像

问题现象: 使用客户端上传镜像, 报如下所示错误:

“Not allow to login、upload or download image”

问题原因：用户大批量并发上传镜像或者攻击服务，系统把用户拉黑，用户无法登录和上传下载镜像。

解决方法：请在30分钟之后重新尝试。

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。

8.6.3 为什么通过页面上上传镜像失败？

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。

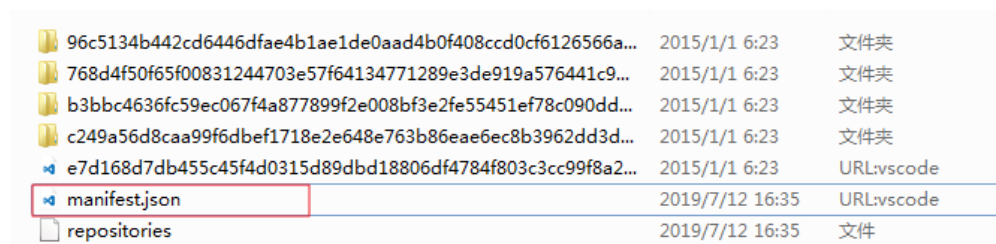
镜像地址各个部分的含义如下，最后的tag（版本号）可省略，如果省略则表示latest版本，其余部分均不可省略，且不可多余。

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。

图 8-4 解压后的文件



96c5134b442cd6446dfae4b1ae1de0aad4b0f408ccd0cf6126566a...	2015/1/1 6:23	文件夹
768d4f50f65f00831244703e57f64134771289e3de919a576441c9...	2015/1/1 6:23	文件夹
b3bbc4636fc59ec067f4a877899f2e008bf3e2fe55451ef78c090dd...	2015/1/1 6:23	文件夹
c249a56d8caa99f6dbef1718e2e648e763b86eae6ec8b3962dd3d...	2015/1/1 6:23	文件夹
e7d168d7db455c45f4d0315d89dbd18806df4784f803c3cc99f8a2...	2015/1/1 6:23	URL:vscode
manifest.json	2019/7/12 16:35	URL:vscode
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，然后再使用页面上上传。

不允许用户上传镜像

问题现象：使用页面上上传镜像，报如下所示错误：

“Not allow to login、upload or download image”

问题原因：用户大批量并发上传镜像或者攻击服务，系统把用户拉黑，用户无法登录和上传下载镜像。

解决方法：请在30分钟之后重新尝试。

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。

8.6.4 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。以下2种方式任选一种（注意：请将镜像仓库地址修改为实际镜像仓库地址）：
 - /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 docker start**。
- 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。

不允许用户下载镜像

问题现象: 使用客户端下载镜像，报如下所示错误：

“Not allow to login、upload or download image”

问题原因: 用户大批量并发上传镜像或者攻击服务，系统把用户拉黑，用户无法登录和上传下载镜像。

解决方法: 请在30分钟之后重新尝试。

8.6.5 API 调用异常

调用SWR的API的url里如果带镜像名称，并且镜像名称带/时，将/替换成\$调用。

例如有个名为a/b的镜像属于c组织：

调用查看镜像信息接口时需要把：`GET /v2/manage/namespaces/c/repos/a/b` 替换为：`GET /v2/manage/namespaces/c/repos/a$b`

8.7 其他

8.7.1 为什么 CCE 工作负载拉取 SWR 内的镜像异常，且提示为“未登录”？

当CCE工作负载无法正常拉取SWR的镜像，且提示“未登录”时，请排查该工作负载的yaml文件中是否存在“`imagePullSecrets`”字段，且`name`参数值需固定为`default-secret`。

示例：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          imagePullPolicy: Always
          name: nginx
      imagePullSecrets:
        - name: default-secret
```

8.7.2 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v5，使用**docker images**命令查询SIZE为**22.8MB**：

```
$ docker images
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
nginx                v5          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	下载指令	更新时间
v5	9.5 MB	docker pull swr.cn-north	2024/05/10 18:10:16 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	下载指令	更新时间
v5	23.2 MB	docker pull swr.cn-north	2024/05/11 09:29:01 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。

8.7.3 SWR 私有镜像最多可以共享给多少个租户？

500个。