Amazon ECR

用户指南

API 版本 2015-09-21

Amazon ECR: 用户指南

Table of Contents

什么是 Amazon Elastic Container Registry？

Amazon Elastic Container Registry (Amazon ECR) 是一项托管 AWS Docker 注册表服务，安全、可扩展且可靠。通过使用 AWS IAM，Amazon ECR 支持具有基于资源的权限的私有 Docker 存储库，以便特定用户或 Amazon EC2 实例可以访问存储库和映像。开发人员可以使用 Docker CLI 推送、拉取和管理映像。

Amazon ECR 的组成部分

Amazon ECR 包含以下组件：

注册表

每个 AWS 账户均提供 Amazon ECR 注册表；您可以在注册表中创建映像存储库，并在其中存储映像。有关更多信息，请参阅Amazon ECR 注册表 (p. 10)。

授权令牌

Docker 客户端必须作为 AWS 用户向 Amazon ECR 注册表进行身份验证，然后才能推送和拉取映像。AWS CLI get-login 命令可为您提供传递给 Docker 的身份验证凭证。有关更多信息，请参阅注册表身份验证 (p. 10)。

存储库

Amazon ECR 映像存储库包含您的 Docker 映像。有关更多信息，请参阅Amazon ECR 存储库 (p. 12)。

存储库策略

您可以通过存储库策略来控制对存储库及其中的映像的访问。有关更多信息，请参阅 Amazon ECR 存储库策略 (p. 14)。

映像

您可以对存储库推送和拉取 Docker 映像。这些映像可以在开发系统中本地使用，也可以在 Amazon ECS 任务定义中使用。有关更多信息，请参阅在 Amazon ECS 中使用 Amazon ECR 映像 (p. 26)。

如何开始使用 Amazon ECR

要使用 Amazon ECR，必须设置以安装 AWS Command Line Interface 和 Docker。有关更多信息，请参阅 Amazon ECR 的设置 (p. 2) 和 Docker 基本知识 (p. 5)。

设置完毕后，您便基本上完成了 Amazon ECR 入门 (p. 9) 教程。

Amazon ECR 的设置

如果已注册 AWS 并已在使用 Amazon Elastic Container Service (Amazon ECS)，那么您与使用 Amazon ECR 已近在咫尺。这两种服务的设置过程相似，因为 Amazon ECR 是 Amazon ECS 的扩展。要在 Amazon ECR 中使用 AWS CLI，必须使用支持最新 Amazon ECR 功能的 AWS CLI 版本。如果在 AWS CLI 中没有看到对 Amazon ECR 功能的支持，可以升级到最新版本。有关更多信息，请参阅 http://www.amazonaws.cn/cli/。

要开始设置 Amazon ECR，请完成以下任务。如果您已完成以下任何步骤，可以将其跳过并继续安装自定义 AWS CLI。

1. 注册 AWS (p. 2)
2. 创建 IAM 用户 (p. 2)
3. 安装 AWS CLI (p. 3)

注册 AWS

当您注册 AWS 时，您的 AWS 账户会自动注册所有服务，包括 Amazon ECR。您只需为使用的服务付费。

如果您已有 AWS 账户，请跳到下一个任务。如果您还没有 AWS 账户，请使用以下步骤创建。

创建 AWS 账户

1. 打开 http://www.amazonaws.cn/，然后选择 Create an AWS Account (创建 AWS 账户)。

   Note

   如果您之前曾使用 AWS 账户根用户 凭证登录 AWS 管理控制台，请选择 Sign in to a different account (登录其他账户)。如果您之前曾使用 IAM 凭证登录控制台，请选择 Sign-in using root account credentials (使用根账户凭证登录)。然后选择 Create a new AWS account (创建新的 AWS 账户)。
2. 按照联机说明操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

请记下您的 AWS 账号，因为在下一个任务中您会用到它。

创建 IAM 用户

AWS 中的服务（例如 Amazon ECR）要求您在访问时提供凭证，以便服务可以确定您是否有权限访问其资源。控制台要求您的密码。您可以为您的 AWS 账户创建访问密钥以访问命令行界面或 API。但是，我们不建议您使用 AWS 账户的凭证访问 ，而建议您改用 AWS (AWS Identity and Access Management)。创建 IAM 用户，然后将该用户添加到具有管理权限的 IAM 组或授予此用户管理权限。然后，您就可以使用专门的 URL 和该 AWS 用户的凭证来访问 。

如果您已注册 AWS 但尚未为自己创建一个 IAM 用户，则可以使用 IAM 控制台自行创建。

为您自己创建一个 IAM 用户并将该用户添加到管理员组

1. 使用 AWS 账户电子邮件地址和密码，以 AWS 账户根用户 身份登录到 IAM 控制台 (https://console.aws.amazon.com/iam/)。

Note

强烈建议您遵守以下使用 **Administrator** IAM 用户的最佳实践，妥善保存根用户凭证。只在执行少数账户和服务管理任务时才作为根用户登录。

2. 在控制台的导航窗格中，选择 Users (用户)，然后选择 Add user (添加用户)。
3. 对于 User name (用户名)，键入 **Administrator**。
4. 选中 AWS 管理控制台 access (AWS 管理控制台访问) 旁边的复选框，选择 Custom password (自定义密码)，然后在文本框中键入新用户的密码。您可以选择 Require password reset (需要重置密码) 以强制用户在下次登录时创建新密码。
5. 选择下一步: 权限。
6. 在设置权限页面上，选择将用户添加到组。
7. 选择 Create group。
8. 在 Create group (创建组) 对话框中，对于 Group name (组名称)，键入 **Administrators**。
9. 对于 Filter policies (筛选策略)，选中 AWS managed - job function (AWS 托管 - 工作职能) 的复选框。
10. 在策略列表中，选中 AdministratorAccess 的复选框。然后选择 Create group。
11. 返回到组列表中，选中您的新组所对应的复选框。如有必要，选择 Refresh 以在列表中查看该组。
12. 选择 Next: Tags (下一步: 标签) 通过以键值对的形式附加标签来向用户添加元数据。
13. 选择 Next: Review 以查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择 Create user。

您可使用此相同的流程创建更多的组和用户，并允许您的用户访问 AWS 账户资源。要了解有关使用策略限制用户对特定 AWS 资源的权限的信息，请参阅访问管理和示例策略。

要以该新 IAM 用户的身份登录，请从 AWS 控制台注销，然后使用以下 URL，其中 your_aws_account_id 是您不带连字符的 AWS 账号（例如，如果您的 AWS 账号是 `1234-5678-9012`，则您的 AWS 账户 ID 是 `123456789012`）：

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

输入您刚创建的 IAM 用户名和密码。登录后，导航栏显示 your_user_name @ your_aws_account_id。

如果您不希望您的登录页面 URL 包含 AWS 账户 ID，可以创建账户别名。从 IAM 控制面板中，选择 Create Account Alias (创建账户别名)，然后输入一个别名，例如您的公司名称。要在创建账户别名后登录，请使用以下 URL：

```
https://your_account_alias.signin.aws.amazon.com/console/
```

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM users sign-in link (IAM 用户登录链接) 下进行检查。

有关 IAM 的更多信息，请参阅 AWS Identity and Access Management 用户指南。

安装 AWS CLI

可以使用 AWS 命令行工具，在系统的命令行中发出命令以执行 Amazon ECS 和 AWS 任务。与使用控制台相比，此方法更快、更方便。命令行工具也非常适用于构建执行 AWS 任务的脚本。

要在 Amazon ECR 中使用 AWS CLI，请安装最新的 AWS CLI 版本（AWS CLI 中从 1.9.15 版本开始提供 Amazon ECR 功能）。可以使用 aws --version 命令查看 AWS CLI 版本。有关安装 AWS CLI 或升级到最新版本的信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

安装 Docker

要在 Amazon ECR 中使用 Docker CLI，必须先在系统上安装 Docker。有关安装 Docker 并熟悉工具的信息，请参阅 Docker 基本知识 (p. 5)。

Docker 基本知识

Docker 是一项可让您构建、运行、测试和部署基于 Linux 容器的分布式应用程序的技术。 Amazon ECR 是一项托管 AWS Docker 注册表服务。客户可以使用熟悉的 Docker CLI 推送、拉取和管理映像。有关 Amazon ECR 产品详细信息、特色客户案例研究和常见问题，请参阅 Amazon Elastic Container Registry 产品详细信息页面。

本指南中的文档假定读者已基本了解 Docker 是什么及其工作方式。有关 Docker 的更多信息，请参阅 Docker 是什么？和 Docker 用户指南。

主题

安装 Docker

Docker 适用于许多不同的操作系统，包括大多数现代 Linux 分发版（如 Ubuntu）甚至 Mac OSX 和 Windows。有关如何在特定的操作系统上安装 Docker 的更多信息，请转到 Docker 安装指南。

您甚至无需本地开发系统即可使用 Docker。如果您已使用 Amazon EC2，则可启动 Amazon Linux 实例并安装 Docker 以开始使用。

在 Amazon Linux 实例上安装 Docker

1. 使用 Amazon Linux AMI 启动实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的启动实例。
2. 连接到您的实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的 Linux 实例。
3. 更新实例上已安装的程序包和程序包缓存。

```
[ec2-user ~]$ sudo yum update -y
```

4. 安装最新的 Docker Community Edition 程序包。

```
[ec2-user ~]$ sudo yum install -y docker
```

5. 启动 Docker 服务。

```
[ec2-user ~]$ sudo service docker start
Starting cgconfig service:                                 [  OK  ]
Starting docker:                                        [  OK  ]
```

6. 将 ec2-user 添加到 docker 组，以便您能够执行 Docker 命令，而无需使用 sudo。

```
[ec2-user ~]$ sudo usermod -a -G docker ec2-user
```

7. 退出，再重新登录以接受新的 docker 组权限。
8. 验证 ec2-user 是否能在没有 sudo 的情况下运行 Docker 命令。

```
[ec2-user ~]$ docker info
```

```
Containers: 2
Images: 24
Storage Driver: devicemapper
 Pool Name: docker-202:1-263460-pool
 Pool Blocksize: 65.54 kB
 Data file: /var/lib/docker/devicemapper/devicemapper/data
 Metadata file: /var/lib/docker/devicemapper/devicemapper/metadata
 Data Space Used: 702.3 MB
 Data Space Total: 107.4 GB
 Metadata Space Used: 1.864 MB
 Metadata Space Total: 2.147 GB
 Library Version: 1.02.89-RHEL6 (2014-09-01)
Execution Driver: native-0.2
Kernel Version: 3.14.27-25.47.amzn1.x86_64
Operating System: Amazon Linux AMI 2014.09
```

Note

在某些情况下，您可能需要重新启动实例，以便为 `ec2-user` 提供访问 Docker 守护程序的权限。如果您看到以下错误消息，请尝试重启您的实例：

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

创建 Docker 映像

在此部分中，您将创建简单 PHP Web 应用程序的 Docker 映像，并在本地系统或 EC2 实例上测试此映像。

创建 PHP Web 应用程序的 Docker 映像

1. 安装 git 并使用它将简单 PHP 应用程序从 GitHub 存储库克隆到系统中。

   a. 安装 git。

   ```
   [ec2-user ~]$ sudo yum install -y git
   ```

   b. 将简单 PHP 应用程序克隆到系统中。

   ```
   [ec2-user ~]$ git clone https://github.com/awslabs/ecs-demo-php-simple-app
   ```

2. 将目录更改为 `ecs-demo-php-simple-app` 文件夹。

   ```
   [ec2-user ~]$ cd ecs-demo-php-simple-app
   ```

3. 在此文件中检查 Dockerfile。Dockerfile 是一个清单文件，描述了用于 Docker 映像的基本映像以及要安装的项目以及在此项目上运行的内容。有关 Dockerfile 的更多信息，请转到 Dockerfile 参考。

   ```
   [ec2-user ecs-demo-php-simple-app]$ cat Dockerfile
   FROM ubuntu:12.04

   # Install dependencies
   RUN apt-get update -y
   RUN apt-get install -y git curl apache2 php5 libapache2-mod-php5 php5-mcrypt php5-mysql

   # Install app
   RUN rm -rf /var/www/*
   ADD src /var/www
   ```

```
# Configure apache
RUN a2enmod rewrite
RUN chown -R www-data:www-data /var/www
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D",  "FOREGROUND"]
```

此 Dockerfile 使用 Ubuntu 12.04 映像。`RUN` 指令将更新程序包缓存，为 Web 服务器和 PHP 支持安装一些软件包，然后将您的 PHP 应用程序添加到 Web 服务器的文档根目录。`EXPOSE` 指令在容器上公开端口 80，`CMD` 指令启动 Web 服务器。

4. 从您的 Dockerfile 构建 Docker 映像并在默认 Amazon ECR 注册表中将该映像标记为 `amazon-ecs-sample`。将 *aws_account_id* 替换为您的 AWS 账户 ID。

   Note

   Docker 的某些版本可能需要在以下命令中使用 Dockerfile 完整路径，而不是所示的相对路径。

```
[ec2-user ecs-demo-php-simple-app]$ docker build -t aws_account_id.dkr.ecr.us-
east-1.amazonaws.com/amazon-ecs-sample .
```

5. 运行 docker images 以验证是否已正确创建映像以及映像名称是否包含可推送到的存储库（在此示例中，您的 Amazon ECR 注册表）。

```
[ec2-user ecs-demo-php-simple-app]$ docker images
REPOSITORY                                                          TAG
 IMAGE ID            CREATED             VIRTUAL SIZE
aws_account_id.dkr.ecr.us-east-1.amazonaws.com/amazon-ecs-sample   latest
 8df953fe88f7        27 minutes ago      260.8 MB
ubuntu                                                              12.04
 2a7a952931ec        3 weeks ago         136.1 MB
```

6. 运行新构建的映像。`-p 80:80` 选项将容器上公开的端口 80 映射到主机系统上的端口 80。有关 docker run 的更多信息，请转到 Docker 运行参考。

```
[ec2-user ecs-demo-php-simple-app]$ docker run -p 80:80 aws_account_id.dkr.ecr.us-
east-1.amazonaws.com/amazon-ecs-sample
apache2: Could not reliably determine the server's fully qualified domain name, using
 172.17.0.2 for ServerName
```

   Note

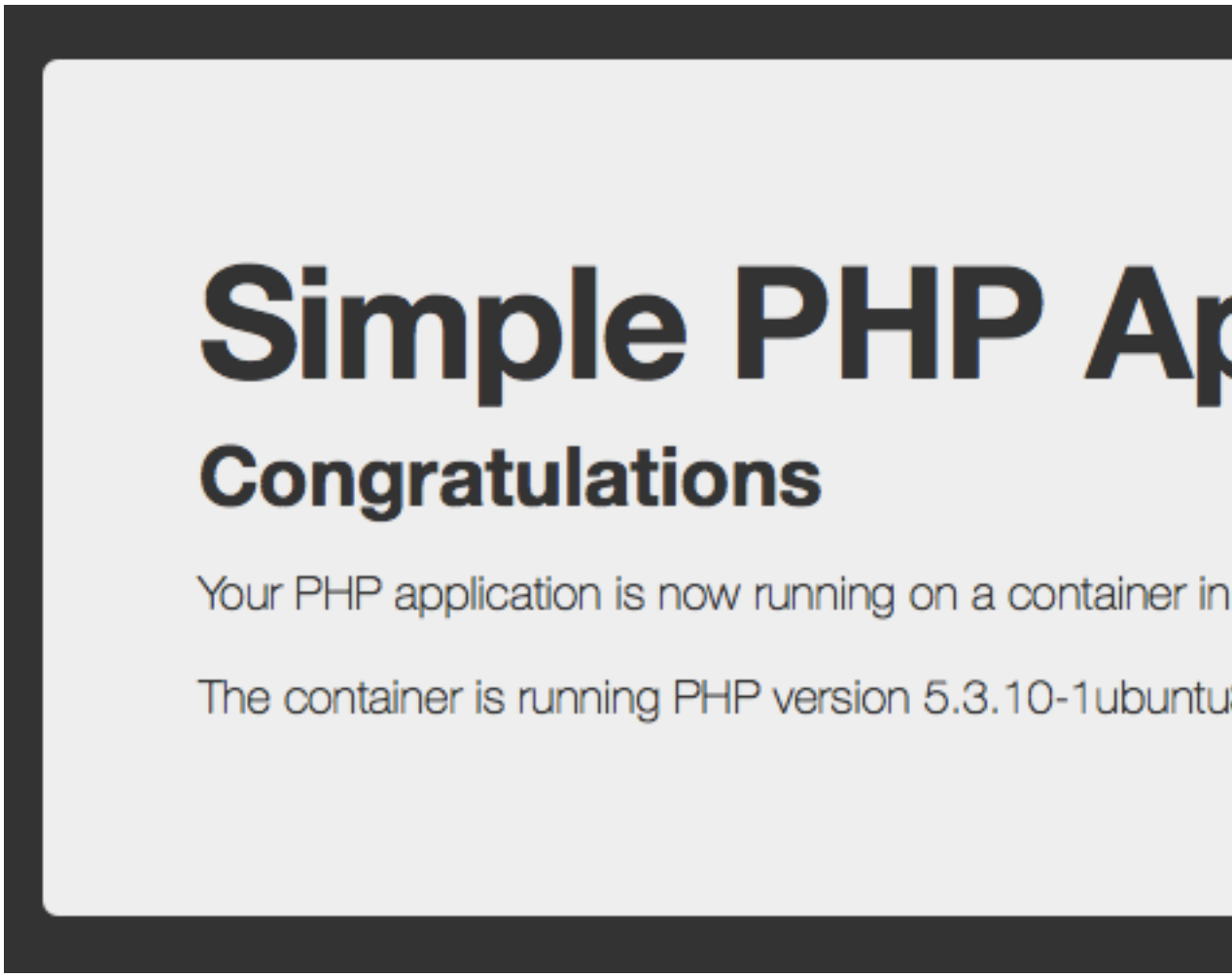
   来自 Apache Web 服务器的输出将显示在终端窗口中。您可以忽略“`Could not reliably determine the server's fully qualified domain name`”消息。

7. 打开浏览器并指向正在运行 Docker 并托管您的容器的服务器。

   - 如果您使用的是 EC2 实例，这将是服务器的 Public DNS 值，此值与您用于通过 SSH 连接到实例的地址相同。确保实例的安全组允许端口 80 上的入站流量。
   - 如果您正在本地运行 Docker，可将您的浏览器指向 http://localhost/。
   - 如果您正在 Windows 或 Mac 计算机上使用 docker-machine，请使用 docker-machine ip 命令查找托管 Docker 的 VirtualBox VM 的 IP 地址，并将 *machine-name* 替换为您正在使用的 Docker 计算机的名称。

```
$ docker-machine ip machine-name
192.168.59.103
```

您应看到正在运行简单 PHP 应用程序的网页。



8. 通过键入 Ctrl+c 来停止 Docker 容器。

后续步骤

现在，您已创建并测试映像，可以执行 Amazon ECR 入门 (p. 9)或将 AWS CLI 与 Amazon ECR 结合使用 (p. 47)中的过程来将映像推送至您的 Amazon ECR 注册表。

Amazon ECR 入门

通过在 Amazon ECR 控制台中创建存储库开始使用 Amazon Elastic Container Registry (Amazon ECR)。Amazon ECR 首次运行向导可以引导您完成开始创建第一个存储库的过程。

> **Important**
>
> 开始之前，请确保您已完成Amazon ECR 的设置 (p. 2)中的步骤。

配置存储库

存储库是您在 Amazon ECR 中存储 Docker 镜像的位置。当您在 Amazon ECR 中推送或拉取映像时，您将指定注册表和存储库位置以告知 Docker 将映像推送到哪个位置或从哪个位置拉取映像。

1. 打开 https://console.amazonaws.cn/ecs/home#/repositories 上的 Amazon ECS 控制台存储库页面。
2. 对于 Repository name，输入存储库的唯一名称并选择 Next step。

构建、标记和推送 Docker 映像

在向导的此部分中，您使用 Docker CLI 标记现有本地映像（您从 Dockerfile 构建或从另一个注册表中拉取的映像，例如 Docker Hub），然后将标记的映像推送到 Amazon ECR 注册表。

1. 检索 docker login 命令，此命令可用于通过将控制台中的 aws ecr get-login 命令粘贴到终端窗口中来对注册表验证 Docker 客户端。

   > **Note**
   >
   > AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本 (17.06 或更高版本)，我们建议使用 1.11.91 或更高版本。您可以使用 aws --version 命令查看 AWS CLI 的版本。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

2. 运行上一步中返回的 docker login 命令。此命令提供一个在 12 小时内有效的授权令牌。

   > **Important**
   >
   > 在执行此 docker login 命令时，进程列表 (ps -e) 显示中将为系统上的其他用户显示命令字符串。由于 docker login 命令包含验证凭证，因此系统上的其他用户可按此方式查看凭证并使用这些凭证来获取对存储库的推送和拉取访问权会带来风险。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p` *`password`* 选项并在系统提示时输入密码来以交互方式登录。

3. (可选) 如果您有要让映像推送的 Dockerfile，请通过将控制台中的 docker build 命令粘贴到终端窗口来为新存储库构建并标记映像。确定您与您的 Dockerfile 在同一目录中。
4. 通过将控制台中的 docker tag 命令粘贴到终端窗口中来为 ECR 注册表和新存储库标记映像。控制台命令假定您已在上一步中从 Dockerfile 构建映像；如果您未从 Dockerfile 构建映像，请将 *`repository`*`:latest` 的第一个实例替换为要推送的本地映像的映像 ID 或映像名。
5. 通过将 docker push 命令粘贴到终端窗口中来将新标记的映像推送到 ECR 存储库。
6. 选择完成。

Amazon ECR 注册表

您可以使用 Amazon ECR 注册表在一个可用性和可扩展性都非常高的架构中托管您的映像，从而安全可靠地为应用程序部署容器。您可以使用注册表管理映像存储库和 Docker 映像。每个 AWS 账户提供单个（默认）Amazon ECR 注册表。

注册表概念

- 您的默认注册表的 URL 为 `https://aws_account_id.dkr.ecr.region.amazonaws.com`。
- 默认情况下，您有在默认注册表中读取和写入您所创建的存储库和映像的权限。
- 您必须为 Docker 客户端授予注册表权限，以便使用 docker push 和 docker pull 命令对该注册表中的存储库执行推送和拉取映像操作。有关更多信息，请参阅注册表身份验证 (p. 10)。
- 可通过 IAM 用户访问策略及存储库策略对存储库加以控制。

注册表身份验证

可以使用 AWS 管理控制台、AWS CLI 或 AWS 开发工具包来创建和管理存储库。也可以使用这些方法对映像执行某些操作，例如列出或删除映像。这些客户端使用标准 AWS 身份验证方法。尽管在技术上可以使用 Amazon ECR API 推送和拉取映像，但您更有可能使用 Docker CLI（或特定语言的 Dockerf 库）。

由于 Docker CLI 不支持标准的 AWS 身份验证方法，必须采用其他方式验证 Docker 客户端的身份。这样 Amazon ECR 才能了解是谁在请求推送或拉取映像。如果您正在使用 Docker CLI，那么可使用 docker login 命令向 Amazon ECR 注册表进行身份验证。可使用 Amazon ECR 提供的授权令牌，该令牌在 12 小时内有效。GetAuthorizationToken API 操作提供采用 base64 编码的授权令牌，其中包含用户名 (AWS) 和密码，解码后可在 docker login 命令中使用。但是，AWS CLI 中提供了更简单的 get-login 命令（检索令牌、解码并将其转换为 docker login 命令）。

使用 get-login 对 Amazon ECR 注册表验证 Docker

> **Note**
>
> AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本 (17.06 或更高版本)，我们建议使用 1.11.91 或更高版本。您可以使用 aws --version 命令查看 AWS CLI 的版本。

1. 运行 aws ecr get-login 命令。以下示例适用于与创建请求的账户关联的默认注册表。要访问其他账户注册表，请使用 `--registry-ids aws_account_id` 选项。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。有关更多信息，请参阅 AWS CLI Command Reference 中的 get-login。

```
aws ecr get-login
```

输出:

```
docker login -u AWS -p password -e none https://aws_account_id.dkr.ecr.us-
east-1.amazonaws.com
```

Important

如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

结果输出是 docker login 命令，此命令可用于对 Amazon ECR 注册表验证 Docker 客户端。

2. 将 docker login 命令复制并粘贴到终端，授权您的 Docker CLI 访问注册表。此命令提供一个授权令牌，此令牌在 12 小时内对指定注册表有效。

Note

如果使用的是 Windows PowerShell，复制并粘贴这样的长字符串将不起作用。请使用以下命令。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。

```
Invoke-Expression -Command (aws ecr get-login)
```

Important

在执行此 docker login 命令时，进程列表 (ps -e) 显示中将为系统上的其他用户显示命令字符串。由于 docker login 命令包含验证凭证，因此系统上的其他用户可按此方式查看凭证并使用这些凭证来获取对存储库的推送和拉取访问权会带来风险。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p` *`password`* 选项并在系统提示时输入密码来以交互方式登录。

HTTP API 身份验证

Amazon ECR 支持 Docker 注册表 HTTP API。但是，由于 Amazon ECR 属于私有注册表，因此您必须为每个 HTTP 请求提供授权令牌。您可以通过使用 curl 的 `-H` 选项来添加 HTTP 授权标头，以传递由 get-authorization-token AWS CLI 命令提供的授权令牌。

使用 Amazon ECR HTTP API 进行身份验证

1. 使用 AWS CLI 检索授权令牌并将其设置为环境变量。

```
TOKEN=$(aws ecr get-authorization-token --output text --query
 'authorizationData[].authorizationToken')
```

2. 要向 API 进行身份验证，可将 `$TOKEN` 变量传递到 curl 命令的 `-H` 选项。例如，以下命令会列出 Amazon ECR 存储库中的映像标签。有关更多信息，请参阅 Docker 注册表 HTTP API 参考文档。

```
curl -i -H "Authorization: Basic $TOKEN" https://012345678910.dkr.ecr.us-
east-1.amazonaws.com/v2/amazonlinux/tags/list
```

输出:

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive

{"name":"amazonlinux","tags":["2017.09","latest"]}
```

Amazon ECR 存储库

Amazon Elastic Container Registry (Amazon ECR) 提供了 API 操作来创建、监控和删除映像存储库，并设置权限以管理谁可以访问存储库。可以在 Amazon ECR 控制台的 Repositories (存储库) 部分中执行相同的操作。Amazon ECR 还集成了 Docker CLI，以便您在开发环境与存储库之间推送和拉取映像。

主题

存储库概念

- 默认情况下，您的账户可以读取和写入默认注册表中的存储库 (*aws_account_id*.dkr.ecr.*region*.amazonaws.com)。但是，IAM 用户需拥有调用 Amazon ECR API 的权限才能从您的存储库中推送或拉取映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问；有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。
- 可通过 IAM 用户访问策略及存储库策略对存储库加以控制。有关更多信息，请参阅 Amazon ECR 存储库策略 (p. 14)。
- 存储库名称可支持命名空间，您可以使用命名空间分组相似的存储库。例如，如果多个团队使用相同的注册表，团队 A 可以使用 `team-a` 命名空间，团队 B 可以使用 `team-b` 命名空间。每个团队可以拥有自己的名为 `web-app` 的映像，因为它们均以团队命名空间作为前缀，所有两个映像可以同时使用，不会互相干扰。团队 A 的映像应称为 `team-a/web-app`，团队 B 的映像则称为 `team-b/web-app`。

创建存储库

在将 Docker 映像推送到 Amazon ECR 之前，必须先创建用于存储映像的存储库。您可以使用 AWS 管理控制台、AWS CLI 和 AWS 开发工具包来创建 Amazon ECR 存储库。

创建存储库

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择要创建您的存储库的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择 Create repository。
5. 对于 Repository configuration (存储库配置)，输入存储库的唯一名称并选择 Create repository (创建存储库)。
6. （可选）选择创建的存储库，并选择 View push commands (查看推送命令) 以查看将映像推送到新存储库的步骤。

   a. 检索 docker login 命令，此命令可用于通过将控制台中的 aws ecr get-login 命令粘贴到终端窗口中来对注册表验证 Docker 客户端。

Note

AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本 (17.06 或更高版本)，我们建议使用 1.11.91 或更高版本。您可以使用 aws --version 命令查看 AWS CLI 的版本。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

b. 运行上一步中返回的 docker login 命令。此命令提供一个在 12 小时内有效的授权令牌。

Important

在执行此 docker login 命令时，进程列表 (ps -e) 显示中将为系统上的其他用户显示命令字符串。由于 docker login 命令包含验证凭证，因此系统上的其他用户可按此方式查看凭证并使用这些凭证来获取对存储库的推送和拉取访问权会带来风险。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p` *`password`* 选项并在系统提示时输入密码来以交互方式登录。

c. (可选) 如果您有要让映像推送的 Dockerfile，请通过将控制台中的 docker build 命令粘贴到终端窗口来为新存储库构建并标记映像。确定您与您的 Dockerfile 在同一目录中。

d. 通过将控制台中的 docker tag 命令粘贴到终端窗口中来为 ECR 注册表和新存储库标记映像。控制台命令假定您已在上一步中从 Dockerfile 构建映像；如果您未从 Dockerfile 构建映像，请将 *`repository`*`:latest` 的第一个实例替换为要推送的本地映像的映像 ID 或映像名。

e. 通过将 docker push 命令粘贴到终端窗口中来将新标记的映像推送到 ECR 存储库。

f. 选择完成。

查看存储库信息

创建存储库后，可以在 AWS 管理控制台中查看其信息：

- 存储库中存储了哪些映像
- 映像是否有标签
- 映像的标签
- 映像的 SHA 摘要
- 映像的大小（以 MiB 为单位）
- 映像推送到存储库的时间

Note

从 Docker 版本 1.9 开始，在将映像推送至 V2 版本的 Docker 注册表之前，Docker 客户端会压缩映像的分层。docker images 命令的输出显示未压缩映像的大小，因此，实际返回的映像大小可能会超过 AWS 管理控制台中显示的映像大小。

查看存储库信息

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要查看的存储库的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择要查看的存储库。
5. 在 Repositories : ***`repository_name`*** (存储库: repository_name) 页面上，使用导航栏查看有关映像的信息。
   - 选择 Images (映像) 以查看有关存储库中映像的信息。如果您要删除的映像没有标签，您可以选择要删除的存储库左侧的框，然后选择 Delete。有关更多信息，请参阅删除映像 (p. 27)。

- 选择 Permissions (权限) 以查看适用于存储库的存储库策略。有关更多信息，请参阅Amazon ECR 存储库策略 (p. 14)。
- 选择 Lifecycle Policy (生命周期策略) 以查看适用于存储库的生命周期策略规则。此处还可查看生命周期事件历史记录。有关更多信息，请参阅 Amazon ECR 生命周期策略 (p. 28)。
- 选择 Tags (标签) 以查看适用于存储库的元数据标签。

删除存储库

如果存储库已使用完毕，您可以删除它。当您在 AWS 管理控制台 中删除存储库时，该存储库中包含的所有映像也将被删除；此操作无法撤消。

删除存储库

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要删除的存储库的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories (存储库) 页面上，选择要删除的存储库，然后选择 Delete (删除)。
5. 在 Delete ***repository_name*** (删除 repository_name) 窗口中，验证是否应删除所选存储库，然后选择 Delete (删除)。

   Important

   选定存储库中的所有映像也会被删除。

Amazon ECR 存储库策略

Amazon ECR 使用基于资源的权限控制访问。基于资源的权限让您可以指定能够访问存储库的用户，以及这些用户可以对该存储库执行的操作。默认情况下，只有存储库所有者有权访问存储库。您可以通过应用策略文档允许他人访问您的存储库。

Important

Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。

主题

- 设置存储库策略声明 (p. 14)
- 删除存储库策略声明 (p. 15)
- Amazon ECR 存储库策略示例 (p. 15)

设置存储库策略声明

您可以在 AWS 管理控制台 中为存储库创建并设置访问策略声明，步骤如下。您可以为每个存储库创建多个策略声明。有关示例策略，请参阅 Amazon ECR 存储库策略示例 (p. 15)。

Important

Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。

设置存储库策略声明

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要对其设置策略声明的存储库的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择要对其设置策略声明的存储库。
5. 在导航窗格中，选择 Permissions (权限)、Edit (编辑)。
6. 在 Edit permissions (编辑权限) 页面上，选择 Add statement (添加声明)。
7. 对于 Statement name (声明名称)，输入声明的名称。
8. 对于 Effect，选择策略声明允许访问还是拒绝访问。
9. 对于 Principal，选择要应用策略声明的用户范围。
   - 通过选中 Everyone (*) (所有人 (*)) 复选框，可以将该声明应用于所有经过身份验证的 AWS 用户。
   - 通过在 AWS account number(s) (AWS 账号) 字段中列出特定的 AWS 账号（例如 111122223333），可以将声明应用到这些账户下的所有用户。
   - 可以在 IAM entities (IAM 实体) 列表下选中角色或用户，然后选择 >> Add (>> 添加) 将这些角色和用户移动到 Selected IAM entities (所选 IAM 实体) 列表中，从而将该声明应用于 AWS 账户下的角色或用户。

     Note

     对于 AWS 管理控制台 中当前不支持的较复杂的存储库策略，您可以使用 set-repository-policy AWS CLI 命令应用此策略。
10. 对于 Action (操作)，从各个 API 操作的列表中选择应当应用策略声明的 Amazon ECR API 操作的范围。
11. 完成后，选择 Save 设置策略。

    Important

    Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。
12. 对要添加的每个存储库策略重复以上步骤。

删除存储库策略声明

如果不再希望将一个现有的策略声明应用至存储库，您可以删除它。

删除存储库策略声明

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要从其中删除策略声明的存储库的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择要从其中删除策略声明的存储库。
5. 在导航窗格中，选择 Permissions (权限)、Edit (编辑)。
6. 在 Edit permissions (编辑权限) 页面上，选择 Delete (删除)。

Amazon ECR 存储库策略示例

以下示例显示了可用于控制用户对 Amazon ECR 存储库的权限的策略声明。

Important

Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。

主题

示例：在您的账户内允许 IAM 用户

以下存储库策略允许您的账户中的 IAM 用户推送和拉取映像。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPushPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::aws_account_id:user/push-pull-user-1",
          "arn:aws:iam::aws_account_id:user/push-pull-user-2"
        ]
      },
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability",
        "ecr:PutImage",
        "ecr:InitiateLayerUpload",
        "ecr:UploadLayerPart",
        "ecr:CompleteLayerUpload"
      ]
    }
  ]
}
```

示例：允许其他账户

以下存储库策略允许特定账户推送映像。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossAccountPush",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::aws_account_id:root"
      },
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
        "ecr:PutImage",
        "ecr:InitiateLayerUpload",
        "ecr:UploadLayerPart",
```

```
        "ecr:CompleteLayerUpload"
      ]
    }
  ]
}
```

以下存储库策略允许所有 AWS 账户拉取映像。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPull",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
}
```

以下存储库策略允许部分 IAM 用户拉取映像（*pull-user-1* 和 *pull-user-2*），并为其他用户提供完全的访问权限 (*admin-user*)。

> **Note**
>
> 对于 AWS 管理控制台 中当前不支持的较复杂的存储库策略，您可以使用 set-repository-policy AWS CLI 命令应用此策略。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::aws_account_id:user/pull-user-1",
          "arn:aws:iam::aws_account_id:user/pull-user-2"
        ]
      },
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid": "AllowAll",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::aws_account_id:user/admin-user"
      },
      "Action": [
        "ecr:*"
      ]
    }
  ]
}
```

示例：拒绝所有

以下存储库策略拒绝所有用户拉取映像。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "DenyPull",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
}
```

标记 Amazon ECR 存储库

为了帮助您管理您的 Amazon ECR 存储库，您可以选择通过标签的形式为每个存储库分配您自己的元数据。本主题介绍标签并说明如何创建标签。

目录

有关标签的基本知识

标签是为 AWS 资源分配的标记。每个标签都包含您定义的一个键 和一个可选值。

标签可让您按各种标准（例如用途、所有者或环境）对 AWS 资源进行分类。这在您具有相同类型的许多资源时会很有用 — 您可以根据分配给资源的标签快速识别特定资源。例如，您可以为账户的 Amazon ECR 存储库定义一组标签以帮助您跟踪每个存储库的拥有者。

我们建议您设计一组满足您的需求的标签键。使用一组连续的标签键，管理资源时会更加轻松。您可以根据添加的标签搜索和筛选资源。

标签对 Amazon ECR 没有任何语义意义，应严格按字符串进行解析。同时，标签不会自动分配至您的资源。您可以修改标签的密钥和值，还可以随时删除资源的标签。您可以将标签的值设为空的字符串，但是不能将其设为空值。如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。如果删除资源，资源的所有标签也会被删除。

可以使用 AWS 管理控制台、AWS CLI 和 Amazon ECR API 处理标签。

如果您使用的是 AWS Identity and Access Management (IAM)，则可以控制 AWS 账户中的哪些用户拥有创建、编辑或删除标签的权限。

标记您的资源

您可以标记新的或现有的 Amazon ECR 存储库。

如果您使用的是 Amazon ECR 控制台，则可以在创建新资源时对其应用标签，或随时在导航窗格上使用 Tags (标签) 选项对现有资源应用标签。

如果您使用的是 Amazon ECR API、AWS CLI 或 AWS 开发工具包，则可以使用 CreateRepository API 操作上的 `tags` 参数对新存储库应用标签，或使用 `TagResource` API 操作对现有资源应用标签。有关更多信息，请参阅 TagResource。

此外，如果无法在存储库创建期间应用标签，则系统将回滚存储库创建过程。这样可确保创建带有标签的存储库，或根本不创建存储库，以及确保任何时候都不创建未标记的存储库。通过在创建时标记存储库，您不需要在存储库创建后运行自定义标记脚本。

标签限制

下面是适用于标签的基本限制：

- 每个存储库的最大标签数 – 50
- 对于每个存储库，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大键长度 – 128 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 – 256 个 Unicode 字符（采用 UTF-8 格式）
- 如果您的标记方案针对多个服务和资源使用，请记得其他服务可能对允许使用的字符有限制。通常允许使用的字符包括：可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：+ - = . _ : / @。
- 标签键和值区分大小写。
- 请不要对键或值使用 `aws:` 前缀；它保留供 AWS 使用。您无法编辑或删除带此前缀的标签键或值。具有此前缀的标签不计入每个资源的标签数限制。

标记资源以便于计费

您为 Amazon ECR 存储库添加的标签在成本和使用率报告中启用标签后查看成本分配时非常有帮助。有关更多信息，请参阅 Amazon ECR 使用率报告 (p. 53)。

如需查看组合资源的成本，请按具有相同标签键值的资源组织您的账单信息。例如，您可以将特定的应用程序名称用作几个资源的标签，然后组织账单信息，以查看在数个服务中的使用该应用程序的总成本。有关使用标签设置成本分配报告的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的月度成本分配报告。

> **Note**
>
> 如果您已启用报告，则可以在 24 小时后查看当月的数据。

通过控制台使用标签

通过使用 Amazon ECR 控制台，您可以管理与新的或现有的存储库关联的标签。

当您在 Amazon ECR 控制台中选择特定存储库时，可通过在导航窗格中选择 Tags (标签) 来查看标签。

为存储库添加标签

1. 通过以下网址打开 Amazon ECR 控制台：https://console.amazonaws.cn/ecr/。
2. 从导航栏中，选择要使用的区域。

3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择要查看的存储库。
5. 在 Repositories : ***repository_name*** (存储库: repository_name) 页面上，从导航窗格中选择 Tags (标签)。
6. 在 Tags (标签) 页面上，选择 Add tags (添加标签)、Add tag (添加标签)。
7. 在 Edit Tags (编辑标签) 页面上，为每个标签指定键和值，然后选择 Save (保存)。

删除单个资源的标签

1. 通过以下网址打开 Amazon ECR 控制台：https://console.amazonaws.cn/ecr/。
2. 从导航栏中，选择要使用的区域。
3. 在 Repositories 页面上，选择要查看的存储库。
4. 在 Repositories : ***repository_name*** (存储库: repository_name) 页面上，从导航窗格中选择 Tags (标签)。
5. 在 Tags (标签) 页面上，选择 Edit (编辑)。
6. 在 Edit Tags (编辑标签) 页面上，选择要删除的每个标签对应的 Remove (删除)，然后选择 Save (保存)。

通过 AWS CLI 或 API 使用标签

使用以下命令添加、更新、列出和删除资源标签。相应文档提供了示例。

针对 Amazon ECR 资源的标记支持

任务	AWS CLI	API 操作
添加或覆盖一个或多个标签。	tag-resource	TagResource
删除一个或多个标签。	untag-resource	UntagResource

以下示例演示如何使用 AWS CLI 管理标签。

示例 1：标记现有存储库

以下命令标记现有存储库。

```
aws ecr tag-resource --resource-arn
 arn:aws:ecr:region:account_id:repository:repository_name --tags Key=stack,Value=dev
```

示例 2：取消标记现有存储库

以下命令删除现有存储库的标签。

```
aws ecr untag-resource --resource-arn
 arn:aws:ecr:region:account_id:repository:repository_name --tag-keys tag_key
```

示例 3：列出存储库的标签

以下命令列出与现有存储库关联的标签。

```
aws ecr list-tags-for-resource --resource-arn
 arn:aws:ecr:region:account_id:repository:repository_name
```

示例 4：创建存储库并应用标签

以下命令创建一个名为 `test-repo` 的存储库并添加键为 `team`、值为 `devs` 的标签。

```
aws ecr create-repository --repository-name test-repo --tags Key=team,Value=devs
```

映像

Amazon Elastic Container Registry (Amazon ECR) 将 Docker 映像存储在映像存储库中。您可以使用 Docker CLI 从存储库推送和拉取映像。

> **Important**
>
> Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。

主题

推送映像

如果您的开发环境中有可用的 Docker 映像，您可以使用 docker push 命令将其推送到 Amazon ECR 存储库。

> **Important**
>
> Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅 Amazon ECR 托管策略 (p. 42)。

推送 Docker 映像到 Amazon ECR 存储库

1. 向要向其推送映像的 Amazon ECR 注册表验证 Docker 客户端的身份。必须针对每个注册表获得授权令牌，令牌有效期为 12 小时。有关更多信息，请参阅注册表身份验证 (p. 10)。
2. 如果在要推送的注册表中还没有您的映像存储库，请创建它。有关更多信息，请参阅创建存储库 (p. 12)。
3. 识别要推送的映像。运行 docker images 命令列出系统中的映像。

   ```
   docker images
   ```

   在生成的命令输出中，可以通过 *repository:tag* 值或映像 ID 识别映像。
4. 通过要使用的 Amazon ECR 注册表、存储库和可选映像标签名称组合标记您的映像。注册表格式为 *aws_account_id*.dkr.ecr.*region*.amazonaws.com。存储库名称应与您为映像创建的存储库一致。如果省略映像标签，我们将假定标签为 `latest`。

   以下示例使用 ID *e9ae3c220b23* 作为 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/my-web-app 来标记映像。

   ```
   docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app
   ```

5. 使用 docker push 命令推送映像：

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-web-app
```

6. （可选）通过重复Step 4 (p. 22)和Step 5 (p. 23)，向映像应用任何其他标签并将这些标签推送到 Amazon ECR。可以向 Amazon ECR 中的每个映像应用最多 100 个标签。

使用 AWS CLI 重新为映像添加标签

借助 Docker Image Manifest V2 Schema 2 映像，可以使用 put-image 命令的 `--image-tag` 选项重新为现有映像添加标签。无需使用 Docker 拉取或推送映像，即可重新添加标签。对于大型映像，此过程可大大节省重新为映像添加标签所需的网络带宽和时间。

Note

此步骤不适用于 Windows 客户端，因为 AWS CLI 输出文本的方式是由 Shell 解释的。要在 Windows 客户端重新为映像添加标签，请参阅使用 适用于 Windows PowerShell 的 AWS 工具 重新为映像添加标签 (p. 24)。

使用 AWS CLI 重新为映像添加标签

1. 使用 batch-get-image 命令可获取要重新添加标签的映像的映像清单并将其写入环境变量。在此示例中，标签为 *latest*，所在存储库为 *amazonlinux* 的映像的清单被写入环境变量 *MANIFEST* 中。

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
 imageTag=latest --query 'images[].imageManifest' --output text)
```

2. 使用 put-image 命令的 `--image-tag` 选项将映像清单与新标签一起放置到 Amazon ECR 中。在此示例中，映像的标签为 *2017.03*。

   Note

   如果 `--image-tag` 选项在您的 AWS CLI 版本中不可用，请升级到最新版本。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的 安装 AWS 命令行界面。

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-manifest
 "$MANIFEST"
```

3. 验证您的新映像标签是否已附加到您的映像。在以下输出中，映像具有标签 `latest` 和 `2017.03`。

```
aws ecr describe-images --repository-name amazonlinux
```

输出:

```
{
    "imageDetails": [
        {
            "imageSizeInBytes": 98755613,
            "imageDigest":
 "sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a2685dfe6f247227",
            "imageTags": [
                "latest",
                "2017.03"
            ],
            "registryId": "aws_account_id",
            "repositoryName": "amazonlinux",
            "imagePushedAt": 1499287667.0
        }
```

```
    ]
}
```

使用 适用于 Windows PowerShell 的 AWS 工具 重新为映像添加标签

借助 Docker Image Manifest V2 Schema 2 映像，可以使用 适用于 Windows PowerShell 的 AWS 工具 Get-ECRImage cmdlet 的 `-ImageTag` 选项重新为现有映像添加标签。无需使用 Docker 拉取或推送映像，即可重新添加标签。对于大型映像，此过程可大大节省重新为映像添加标签所需的网络带宽和时间。

使用 适用于 Windows PowerShell 的 AWS 工具 重新为映像添加标签

1. 使用 Get-ECRImageBatch cmdlet 获取要重新添加标签的映像的描述，并将该映像写入到环境变量。在此示例中，标签为 *latest*、所在存储库为 *amazonlinux* 的映像被写入环境变量 *$Image* 中。

   Note

   如果您的系统中没有可用的 Get-ECRImageBatch cmdlet，请参阅 适用于 Windows PowerShell 的 AWS 工具 用户指南 中的设置 适用于 Windows PowerShell 的 AWS 工具。

   ```
   $Image = Get-ECRImageBatch -ImageId @{ imageTag="latest" } -RepositoryName amazonlinux
   ```

2. 将该映像的清单写入到 *$Manifest* 环境变量。

   ```
   $Manifest = $Image.Images[0].ImageManifest
   ```

3. 使用 Write-ECRImage cmdlet 的 `-ImageTag` 选项将映像清单与新标签一起放置到 Amazon ECR 中。在此示例中，映像的标签为 *2017.09*。

   ```
   Write-ECRImage -RepositoryName amazonlinux -ImageManifest $Manifest -ImageTag 2017.09
   ```

4. 验证您的新映像标签是否已附加到您的映像。在以下输出中，映像具有标签 `latest` 和 `2017.09`。

   ```
   Get-ECRImage -RepositoryName amazonlinux
   ```

   输出:

   ```
   ImageDigest                                                             ImageTag
   -----------                                                             --------
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 latest
   sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 2017.09
   ```

拉取映像

如果希望运行 Amazon ECR 中可用的 Docker 映像，可以使用 docker pull 命令将其拉取到本地环境。可以从默认注册表或与其他 AWS 账户关联的注册表执行此操作。要在 Amazon ECS 任务定义中使用 Amazon ECR 映像，请参阅在 Amazon ECS 中使用 Amazon ECR 映像 (p. 26)。

Important

Amazon ECR 用户需要先获得调用 `ecr:GetAuthorizationToken` 的权限，然后才能对注册表进行身份验证，并从任何 Amazon ECR 存储库推送或提取任何映像。Amazon ECR 提供一些托管策略来控制不同级别下的用户访问，有关更多信息，请参阅Amazon ECR 托管策略 (p. 42)。

从 Amazon ECR 存储库拉取 Docker 映像

1. 将您的 Docker 客户端验证到要从中拉取映像的 Amazon ECR 注册表。必须针对每个注册表获得授权令牌，令牌有效期为 12 小时。有关更多信息，请参阅注册表身份验证 (p. 10)。
2. (可选)识别要拉取的映像。
   - 可以使用 aws ecr describe-repositories 命令列出注册表中的存储库：

     ```
     aws ecr describe-repositories
     ```

     上述示例注册表包含一个名为 `amazonlinux` 的存储库。
   - 可以使用 aws ecr describe-images 命令描述存储库中的映像：

     ```
     aws ecr describe-images --repository-name amazonlinux
     ```

     上述示例存储库具有带标签 `latest` 和 `2016.09` 的映像，并且映像摘要为 `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`。
3. 使用 docker pull 命令拉取映像。映像名称格式应为 *`registry`*`/`*`repository`*`[:`*`tag`*`]` 以便按标签拉取，或为 *`registry`*`/`*`repository`*`[@`*`digest`*`]` 以便按摘要拉取。

   ```
   docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
   ```

   Important

   如果您收到 *`repository-url`* `not found: does not exist or no pull access` 错误，您可能需要向 Amazon ECR 验证您的 Docker 客户端。有关更多信息，请参阅注册表身份验证 (p. 10)。

容器映像清单格式

Amazon ECR 支持以下容器映像清单格式：

- Docker Image Manifest V2 Schema 1 (与 Docker 版本 1.9 和更早版本配合使用)
- Docker Image Manifest V2 Schema 2 (与 Docker 版本 1.10 和更新版本配合使用)
- Open Container Initiative (OCI) 规范 (v1.0 和更高版本)

对 Docker Image Manifest V2 Schema 2 的支持可提供以下功能：

- 能够为每个映像使用多个标签。
- 支持存储 Windows 容器映像。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的将 Windows 映像推送到 Amazon ECR。

Amazon ECR 映像清单转换

在 Amazon ECR 中推送和拉取映像时，您的容器引擎客户端 (例如 Docker) 将与注册表进行通信以就客户端了解的清单格式以及要用于映像的注册表达成一致。

在使用 Docker 版本 1.9 或更旧版本将映像推送到 Amazon ECR 时，映像清单格式将存储为 Docker Image Manifest V2 Schema 1。在使用 Docker 版本 1.10 或更新版本将映像推送到 Amazon ECR 时，映像清单格式将存储为 Docker Image Manifest V2 Schema 2。

在从 Amazon ECR 按标签 拉取映像时，Amazon ECR 将返回存储在存储库中的映像清单格式。仅当客户端理解该格式时，才会返回该格式。如果客户端不理解所存储的映像清单格式，则 Amazon ECR 会将映像清单转换为客户端能够理解的格式。例如，如果 Docker 1.9 客户端请求的映像清单存储格式为 Docker Image Manifest V2 Schema 2，那么 Amazon ECR 将以 Docker Image Manifest V2 Schema 1 格式返回该清单。下表描述了在按标签 拉取映像时，Amazon ECR 支持的可用转换：

客户端请求的架构	作为 V2 Schema 1 推送到 ECR	作为 V2 Schema 2 推送到 ECR	作为 OCI 推送到 ECR
V2 Schema 1	无需转换	已转换为 V2 Schema 1	已转换为 V2 Schema 1
V2 Schema 2	无可用转换，客户端将回退到 V2 Schema 1	无需转换	已转换为 V2 Schema 2
OCI	无可用转换	已转换为 OCI	无需转换

Important

如果您按摘要 拉取映像，则无可用转换；您的客户端必须了解存储在 Amazon ECR 中的映像清单格式。如果您在 Docker 1.9 或更旧版本的客户端上按摘要请求 Docker Image Manifest V2 Schema 2 映像，则无法拉取映像。有关更多信息，请参阅 Docker 文档中的注册表兼容性。
在此示例中，如果按标签 请求同一映像，Amazon ECR 会将映像清单转换为客户端能够理解的格式。映像拉取成功。

在 Amazon ECS 中使用 Amazon ECR 映像

您可以将 ECR 映像与 Amazon ECS 结合使用，但需要满足以下先决条件：

- 您的容器实例必须至少使用版本 1.7.0 的 Amazon ECS 容器代理。最新版本的经 Amazon ECS 优化的 AMI 在任务定义中支持 ECR 映像。有关更多信息 (包括最新的经 Amazon ECS 优化的 AMI ID)，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器代理版本。
- 您用于容器实例的 Amazon ECS 容器实例角色 (`ecsInstanceRole`) 必须拥有 Amazon ECR 的以下 IAM 策略权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

  如果您对容器实例使用 `AmazonEC2ContainerServiceforEC2Role` 托管策略，则您的角色将具有适当的权限。要检查您的角色是否支持 Amazon ECR，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器实例 IAM 角色。
- 在 ECS 任务定义中，确保对 ECR 映像使用完整的 `registry/repository:tag` 命名。例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*。

删除映像

如果您不再使用映像，可以从存储库中删除它。您可以使用 AWS 管理控制台或 AWS CLI 删除映像。

> Note
>
> 如果您不再使用存储库，可以删除整个存储库以及其中的所有映像。有关更多信息，请参阅删除存储库 (p. 14)。

使用 AWS 管理控制台 删除映像

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要删除的映像的区域。
3. 在导航窗格中，选择 Repositories。
4. 在 Repositories 页面上，选择包含要删除的映像的存储库。
5. 在 Repositories: ***repository_name*** (存储库: repository_name) 页上，选择要删除的映像左侧的框，然后选择 Delete (删除)。
6. 在 Delete image(s) 对话框中，验证选定的映像是否应被删除，然后选择 Delete。

使用 AWS CLI 删除映像

1. 列出存储库中的映像，以便按映像标签或摘要标识映像。

```
aws ecr list-images --repository-name my-repo
```

2. (可选) 通过指定要删除的映像标签来删除映像的任何不需要的标签。

> Note
>
> 删除映像的最后一个标签后，将删除映像。

```
aws ecr batch-delete-image --repository-name my-repo --image-ids imageTag=latest
```

3. 通过指定要删除的映像的摘要来删除映像。

> Note
>
> 在通过引用映像摘要来删除映像时，映像及其所有标签都会被删除。

```
aws ecr batch-delete-image --repository-name my-repo --image-ids
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304c7c2c1a9d6fa3e9de6bf552d
```

Amazon Linux 容器镜像

构建 Amazon Linux 容器映像的软件组件与 Amazon Linux AMI 中包含的软件组件相同。作为 Docker 工作负载的基本镜像，它可用在任何环境中。如果您已经在 Amazon EC2 中针对应用程序使用了 Amazon Linux AMI，就可以轻松使用 Amazon Linux 容器映像将您的应用程序容器化。

可以在本地开发环境中使用 Amazon Linux 容器映像，然后使用 Amazon ECS 将应用程序推送到 AWS 云。有关更多信息，请参阅在 Amazon ECS 中使用 Amazon ECR 映像 (p. 26)。

Amazon Linux 容器映像在 和 Docker Hub 上可用。访问 AWS 开发人员论坛可获得针对 Amazon Linux 容器映像的支持。

从 Docker Hub 拉取 Amazon Linux 容器镜像

1. 使用 docker pull 命令拉取 Amazon Linux 容器映像。

```
docker pull amazonlinux
```

2. （可选）在本地运行容器。

```
docker run -it amazonlinux:latest /bin/bash
```

Amazon ECR 生命周期策略

Amazon ECR 生命周期策略使您能够指定存储库中映像的生命周期管理。生命周期策略是一组规则，其中的每个规则为 Amazon ECR 定义一个操作。这些操作适用于包含前缀为给定字符串的标签的映像。这实现了自动清除未使用的映像，例如根据存在时间或计数使映像过期。您应该预计到在创建生命周期策略之后，受影响的映像将在 24 小时后到期。

主题

生命周期策略模板

在将生命周期策略与存储库关联前，应评估其内容。以下是生命周期策略的 JSON 语法模板。有关生命周期策略示例，请参阅 生命周期策略示例 (p. 32)。

```
{
    "rules": [
        {
            "rulePriority": integer,
            "description": "string",
            "selection": {
                "tagStatus": "tagged"|"untagged"|"any",
                "tagPrefixList": list<string>,
                "countType": "imageCountMoreThan"|"sinceImagePushed",
                "countUnit": "string",
                "countNumber": integer
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Note

仅当 tagPrefixList 为 tagStatus 时，才能使用 tagged 参数。仅当 countUnit 为 countType 时，才能使用 sinceImagePushed 参数。仅当 countNumber 设置为 countType 时，才能使用 imageCountMoreThan 参数。

生命周期策略参数

生命周期策略分成以下几部分：

主题

规则优先级

`rulePriority`

类型：整数

必需：是

设置评估规则的顺序，从低到高。当您向某个生命周期策略添加规则时，必须为每个规则赋予一个唯一的 `rulePriority` 值。但是，在策略中的各规则之间，值不需要顺序。具有 `tagStatus` 值 `any` 的规则必须具有最大的 `rulePriority` 值并且最后被评估。

描述

`description`

类型：字符串

必需：否

（可选）描述生命周期策略中的规则的用途。

标签状态

`tagStatus`

类型：字符串

必需：是

确定您要添加的生命周期策略规则是否为映像指定标签。可接受的选项为 `tagged`、`untagged` 或 `any`。如果指定了 `any`，则所有映像都拥有应用于自身的规则。如果指定了 `tagged`，则还必须指定 `tagPrefixList` 值。如果指定了 `untagged`，则必须省略 `tagPrefixList`。

标签前缀列表

`tagPrefixList`

类型：list[string]

必需：是，仅当 tagStatus 设置为 tagged 时

仅在指定了 "tagStatus": "tagged" 时使用。必须指定要使用生命周期策略对其执行操作的映像标签前缀的逗号分隔列表。例如，如果您的映像标记为 prod、prod1、prod2 等等，则使用标签前缀 prod 来指定所有这些映像。如果指定多个标签，则仅选择具有所有指定标签的映像。

计数类型

countType

类型：字符串

必需：是

指定要应用于映像的计数类型。如果将 countType 设置为 imageCountMoreThan，还要指定 countNumber 来创建一个规则以对存储库中存在的映像数设定限制。如果将 countType 设置为 sinceImagePushed，还要指定 countUnit 和 countNumber 来指定存储库中存在的映像的时间限制。

计数单位

countUnit

类型：字符串

必需：是，仅当 countType 设置为 sinceImagePushed 时

指定计数单位 days 作为时间单位，除此之外，还指定 countNumber 表示天数。

计数数值

countNumber

类型：整数

必需：是

指定计数数量。如果使用的 countType 是 imageCountMoreThan，则该值为您希望在存储库中保留的映像的最大数量。如果使用的 countType 是 sinceImagePushed，则该值为您的映像的最长时间限制。

操作

type

类型：字符串

必需：是

指定操作类型。支持的值为 expire。

生命周期策略评估规则

生命周期策略评估程序负责解析纯文本 JSON 并将其应用于指定存储库中的映像。在创建生命周期策略时，应注意以下规则：

- 使用正好一个或零个规则使一个映像过期。
- 不能用优先级低的规则使符合优先级高的规则标记要求的映像过期。
- 规则绝不能标记由优先级高的规则所标记的映像，但仍可以识别这些映像，就像它们还没有过期一样。
- 规则组中必须包含一组独特的标签前缀。
- 只允许一条规则来选择未标记的映像。
- 始终按照 `pushed_at_time` 来排序映像过期顺序，始终使旧映像先过期，新映像后过期。
- 在使用 `tagPrefixList` 时，如果 `tagPrefixList` 值中的所有 标签与某映像的任一标签匹配，则该映像成功匹配。
- 利用 `countType = imageCountMoreThan`，将基于 `pushed_at_time` 对映像从新到旧排序，所有超过指定计数的映像均过期。
- 利用 `countType = sinceImagePushed`，其 `pushed_at_time` 长于指定天数 (基于 `countNumber`) 的所有映像均过期。

创建生命周期策略预览

生命周期策略预览允许您在执行生命周期策略前查看其对映像存储库的影响。以下过程显示如何创建生命周期策略预览。

使用控制台创建生命周期策略预览

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要对其执行生命周期策略预览的存储库的区域。
3. 在导航窗格中，选择 Repositories 并选择一个存储库。
4. 在 Repositories: ***`repository_name`*** (存储库: repository_name) 页面上，在导航窗格中选择 Lifecycle Policy (生命周期策略)。
5. 在 Repositories: ***`repository_name`***: Lifecycle policy (存储库: repository_name: 生命周期策略) 页面上，选择 Edit test rules (编辑测试规则)、Create rule (创建规则)。
6. 为您的生命周期策略规则输入以下详细信息：
   a. 对于 Rule Priority (规则优先级)，键入规则优先级编号。
   b. 对于 Rule Description (规则描述)，键入对生命周期策略规则的描述。
   c. 对于 Image status (映像状态)，选择 Tagged (已标记)、Untagged (未标记) 或 Any (任意)。
   d. 如果为 Image status (映像状态) 指定了 `Tagged`，则对于 Tag prefixes (标签前缀)，可以选择指定要根据生命周期策略对其执行操作的映像标签列表。如果指定了 `Untagged`，则此字段必须为空。
   e. 对于 Match criteria (匹配条件)，为 Since image pushed (自从推送映像以来) 或 Image count more than (映像计数超过) 选择值（如果适用）。
7. 选择 Save。
8. 通过重复步骤 5-7 来创建其他生命周期策略规则。
9. 要运行生命周期策略预览，请选择 Save and run test (保存并运行测试)。
10. 在 Image matches for test lifecycle rules (测试生命周期规则的映像匹配) 下，查看生命周期策略预览的效果。
11. 如果您对预览结果满意，选择 Apply as lifecycle policy 来使用指定规则创建生命周期策略。

> **Note**
>
> 您应该预计到在创建生命周期策略之后，受影响的映像将在 24 小时后到期。

创建生命周期策略

生命周期策略允许您创建一组可以使未使用的存储库映像过期的规则。以下过程显示如何创建生命周期策略。您应该预计到在创建生命周期策略之后，受影响的映像将在 24 小时后到期。

使用 AWS CLI 创建生命周期策略

1. 获取要为其创建生命周期策略的存储库的 ID：

```
aws ecr describe-repositories
```

2. 创建生命周期策略：

```
aws ecr put-lifecycle-policy [--registry-id <string>] --repository-name <string> --
policy-text <string>
```

使用控制台创建生命周期策略

1. Open the Amazon ECR console at https://console.amazonaws.cn/ecr/repositories.
2. 从导航栏中，选择包含要为其创建生命周期策略的存储库的区域。
3.
4. 在导航窗格中，选择 Repositories 并选择一个存储库。
5. 在 Repositories: ***repository_name*** (存储库: repository_name) 页面上，在导航窗格中选择 Lifecycle Policy (生命周期策略)。
6. 在 Repositories: ***repository_name***: Lifecycle policy (存储库: repository_name: 生命周期策略) 页面上，选择 Create rule (创建规则)。
7. 为您的生命周期策略规则输入以下详细信息：
   a. 对于 Rule Priority (规则优先级)，键入规则优先级编号。
   b. 对于 Rule Description (规则描述)，键入对生命周期策略规则的描述。
   c. 对于 Image status (映像状态)，选择 Tagged (已标记)、Untagged (未标记) 或 Any (任意)。
   d. 如果为 Image status (映像状态) 指定了 `Tagged`，则对于 Tag prefixes (标签前缀)，可以选择指定要根据生命周期策略对其执行操作的映像标签列表。如果指定了 `Untagged`，则此字段必须为空。
   e. 对于 Match criteria (匹配条件)，为 Since image pushed (自从推送映像以来) 或 Image count more than (映像计数超过) 选择值（如果适用）。
8. 选择 Save。

生命周期策略示例

以下是生命周期策略示例，用于显示语法。

主题

按映像存在时间筛选

以下示例显示了用于使超过 14 天的未标记映像过期的策略的生命周期策略语法：

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

按映像计数筛选

以下示例显示了用于只保留一个未标记映像而使其他所有映像均过期的策略的生命周期策略语法：

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Keep only one untagged image, expire all others",
            "selection": {
                "tagStatus": "untagged",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

按多个规则筛选

以下示例在一个生命周期策略中使用多个规则。示例中提供了示例存储库和生命周期策略以及对结果的解释。

示例 A

存储库内容：

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

生命周期策略文本：

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["prod"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["beta"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别用前缀 `prod` 标记的映像。它应该从最旧的映像开始标记，直到剩下一个或零个匹配的映像。它将映像 A 标记为过期。
- 规则 2 识别用前缀 `beta` 标记的映像。它应该从最旧的映像开始标记，直到剩下一个或零个匹配的映像。它将映像 A 和映像 B 均标记为过期。但是，映像 A 已经被规则 1 看到，而如果映像 B 过期，则违犯规则 1，因此跳过映像 B。
- 结果：映像 A 过期。

示例 B

这是与上一示例中相同的存储库，只是改变了规则优先级顺序以阐明结果。

存储库内容：

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

生命周期策略文本：

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
```

```
                "tagPrefixList": ["beta"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["prod"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别用 `beta` 标记的映像。它应该从最旧的映像开始标记，直到剩下一个或零个匹配的映像。它看到所有三个映像，将映像 A 和映像 B 标记为过期。
- 规则 2 识别用 `prod` 标记的映像。它应该从最旧的映像开始标记，直到剩下一个或零个匹配的映像。它看不到任何映像，因为所有可用的映像都已被规则 1 看到，所以规则 2 不会标记额外的映像。
- 结果：映像 A 和 B 过期。

按单个规则中的多个标签筛选

以下示例为单个规则中的多个标签前缀指定生命周期策略语法。示例中提供了示例存储库和生命周期策略以及对结果的解释。

示例 A

在单个规则上指定多个标签前缀时，映像必须匹配所有列出的标签前缀。

存储库内容：

- Image A, Taglist: ["alpha-1"], Pushed: 12 days ago
- Image B, Taglist: ["beta-1"], Pushed: 11 days ago
- Image C, Taglist: ["alpha-2", "beta-2"], Pushed: 10 days ago
- Image D, Taglist: ["alpha-3"], Pushed: 4 days ago
- Image E, Taglist: ["beta-3"], Pushed: 3 days ago
- Image F, Taglist: ["alpha-4", "beta-4"], Pushed: 2 days ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
```

```
                "tagPrefixList": ["alpha", "beta"],
                "countType": "sinceImagePushed",
                "countNumber": 5,
                "countUnit": "days"
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别用 `alpha` 和 `beta` 标记的映像。它看到映像 C 和 F。它应标记超过五天的映像，那将是映像 C。
- 结果：映像 C 过期。

示例 B

以下示例阐述标签不是专用的。

存储库内容：

- Image A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Pushed: 10 days ago
- Image B, Taglist: ["alpha-2", "beta-2"], Pushed: 9 days ago
- Image C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Pushed: 8 days ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["alpha", "beta"],
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别用 `alpha` 和 `beta` 标记的映像。它看到所有映像。它应该从最旧的映像开始标记，直到剩下一个或零个匹配的映像。它将映像 A 和 B 标记为过期。
- 结果：映像 A 和 B 过期。

筛选所有映像

以下生命周期策略示例指定所有具有不同的筛选器的映像。示例中提供了示例存储库和生命周期策略以及对结果的解释。

示例 A

下面显示了适用于所有规则，但只保留一个未标记映像并使其他所有映像过期的策略的生命周期策略语法。

存储库内容：

- Image A, Taglist: ["alpha-1"], Pushed: 4 days ago
- Image B, Taglist: ["beta-1"], Pushed: 3 days ago
- Image C, Taglist: [], Pushed: 2 days ago
- Image D, Taglist: ["alpha-2"], Pushed: 1 day ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别所有映像。它将发现映像 A、B、C 和 D。它应该使除最新映像之外的所有映像过期。它将映像 A、B 和 C 标记为过期。
- 结果：映像 A、B 和 C 过期。

示例 B

以下示例介绍将所有规则类型整合到单个策略中的生命周期策略。

存储库内容：

- Image A, Taglist: ["alpha-", "beta-1", "-1"], Pushed: 4 days ago
- Image B, Taglist: [], Pushed: 3 days ago
- Image C, Taglist: ["alpha-2"], Pushed: 2 days ago
- Image D, Taglist: ["git hash"], Pushed: 1 day ago
- Image E, Taglist: [], Pushed: 1 day ago

```
{
    "rules": [
        {
            "rulePriority": 1,
            "description": "Rule 1",
            "selection": {
                "tagStatus": "tagged",
                "tagPrefixList": ["alpha"],
                "countType": "imageCountMoreThan",
```

```
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 2,
            "description": "Rule 2",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "day",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        },
        {
            "rulePriority": 3,
            "description": "Rule 3",
            "selection": {
                "tagStatus": "any",
                "countType": "imageCountMoreThan",
                "countNumber": 1
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

该生命周期策略的逻辑是：

- 规则 1 识别用 `alpha` 标记的映像。它识别映像 A 和 C。它应该保留最新映像并将其余映像标记为过期。它将映像 A 标记为过期。
- 规则 2 识别未标记的映像。它识别映像 B 和 E。它应该将所有超过 1 天的映像标记为过期。它将映像 B 标记为过期。
- 规则 3 识别所有映像。它识别映像 A、B、C、D 和 E。它应该保留最新映像并将其余映像标记为过期。但是，它无法标记映像 A、B 或 C，因为它们由优先级更高的规则识别。它将映像 D 标记为过期。
- 结果：映像 A、B 和 D 过期。

Amazon ECR IAM 策略和角色

默认情况下，IAM 用户无权创建或修改 Amazon Elastic Container Registry (Amazon ECR) 资源或使用 Amazon ECR API 执行任务。（这意味着，这些用户也无法使用 Amazon ECR 控制台或 AWS CLI 执行这些操作。）要允许 IAM 用户创建或修改资源并执行任务，必须创建 IAM 策略，授予 IAM 用户使用所需的特定资源和 API 操作的权限。然后，将这些策略附加到需要这些权限的 IAM 用户或组。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。有关更多信息，请参阅 IAM 用户指南 中的权限和策略和管理 IAM 策略。

同样，Amazon ECS 容器实例会代表您调用 Amazon ECR API，拉取 Amazon ECS 任务定义中使用的 Docker 映像。它们需要使用您的凭证进行身份验证。在启动容器实例时，通过为容器实例创建 IAM 角色并将该角色与它们关联来执行此身份验证。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器实例 IAM 角色。有关 IAM 角色的详细信息，请参阅 IAM 用户指南 中的 IAM 角色。

入门

IAM 策略必须授予或拒绝使用一个或多个 Amazon ECR 操作的权限。它还必须指定可以用于操作的资源 (可以是所有资源，在某些情况下可以是特定资源)。策略还可以包含应用于资源的条件。

Amazon ECR 部分支持资源级权限。这意味着，对于某些 Amazon ECS API 操作，您无法指定用户可用于该操作的资源。相反，您不得不允许用户将所有资源用于该操作。

主题

策略结构

以下主题说明 IAM 策略的结构。

主题

策略语法

IAM 策略是包含一个或多个语句的 JSON 文档。每个语句的结构如下：

```
{
  "Statement":[{
```

```
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

组成语句的各个元素如下：

- `Effect`：此 effect 可以是 `Allow` 或 `Deny`。默认情况下 IAM 用户没有使用资源和 API 操作的权限，因此，所有请求均会被拒绝。显式允许将覆盖默认规则。显式拒绝将覆盖任何允许。
- `Action`：对其授予或拒绝权限的特定 API 操作。有关更多信息，请参阅针对 Amazon ECR 的操作 (p. 40)。
- `Resource`：受操作影响的资源。有些 Amazon ECR API 操作允许您在策略中包括该操作可以创建或修改的特定资源。要在语句中指定资源，必须使用其 Amazon 资源名称 (ARN)。有关指定 `arn` 值的更多信息，请参阅适用于 Amazon ECR 的Amazon 资源名称 (p. 40)。有关哪些 ARN 支持哪些 API 操作的更多信息，请参阅 Amazon ECR API 操作支持的资源级权限 (p. 44)。如果 API 操作不支持 ARN，请使用 * (星号) 通配符指定操作可以影响所有资源。
- `Condition`：条件是可选的，可控制策略的生效时间。想要了解更多有关为 Amazon ECR 指定条件的信息，请参阅 Amazon ECR 的条件密钥 (p. 42)。

针对 Amazon ECR 的操作

在 IAM 策略语句中，您可以从支持 IAM 的任何服务中指定任何 API 操作。对于 Amazon ECR，请使用以下前缀为 API 操作命名：`ecr:`。例如：`ecr:CreateRepository` 和 `ecr:DeleteRepository`。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": ["ecr:action1", "ecr:action2"]
```

您也可以使用通配符指定多项操作。例如，您可以指定名称以单词“Delete”开头的所有操作，如下所示：

```
"Action": "ecr:Delete*"
```

要指定所有 Amazon ECR API 操作，请使用 * (型号) 通配符，如下所示：

```
"Action": "ecr:*"
```

有关 Amazon ECR 操作的列表，请参阅 Amazon Elastic Container Registry API Reference 中的操作。

适用于 Amazon ECR 的Amazon 资源名称

每个 IAM 策略语句适用于您使用资源的 ARN 指定的资源。

> **Important**
>
> 当前，并非所有 API 操作都支持各个 ARN。我们将在以后添加对其他 API 操作和其他 Amazon ECR 资源的 ARN 的支持。有关哪些 ARN 可以与哪些 Amazon ECR API 操作一起使用的信息，请参阅Amazon ECR API 操作支持的资源级权限 (p. 44)。

ARN 的一般语法如下：

```
arn:aws:[service]:[region]:[account]:resourceType/resourcePath
```

`service`

服务 (例如，`ecr`)。

`region`

资源所在区域 (例如，`us-east-1`)。

`account`

AWS 账户 ID，不包含连字符 (例如，`123456789012`)。

`resourceType`

资源类型 (例如，`instance`)。

`resourcePath`

识别资源的路径。您可以在路径中使用 * (星号) 通配符。

例如，您可以使用特定存储库 (`my-repo`) 的 ARN 在语句中指定它，如下所示：

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

还可以使用 * 通配符指定属于特定账户的所有存储库，如下所示：

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

要指定所有资源，或者如果特定 API 操作不支持 ARN，请在 `Resource` 元素中使用 * 通配符，如下所示：

```
"Resource": "*"
```

下表介绍了 Amazon ECR API 操作使用的每种类型资源的 ARN。

资源类型	ARN
所有 Amazon ECR 资源	`arn:aws:ecr:*`
指定账户在指定地区拥有的所有 Amazon ECR 资源	`arn:aws:ecr:region:account:*`
存储库	`arn:aws:ecr:region:account:repository/repository-name`

许多 Amazon ECR API 操作接受多个资源。要在单个语句中指定多种资源，请使用逗号将它们隔开，如下所示：

```
"Resource": ["arn1", "arn2"]
```

有关更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 资源名称 (ARN) 和 AWS 服务命名空间。

Amazon ECR 的条件密钥

在策略语句中，您可以选择性指定控制策略生效时间的条件。每个条件都包含一个或多个键值对。条件键不区分大小写。我们已经定义了 AWS 范围内的条件键以及其他特定于服务的条件键。

如果指定了多个条件或在单个条件中指定了多个密钥，我们将通过逻辑 AND 操作对其进行评估。如果您在单一条件中指定了一个具有多个值的密钥，我们将通过逻辑 OR 操作对其进行评估。必须匹配所有条件才能授予权限。

在指定条件时，您也可使用占位符。有关更多信息，请参阅 IAM 用户指南 中的策略变量。

Amazon ECR 可实现 AWS 范围内的条件密钥（请参阅可用密钥），但 `aws:SecureTransport` 条件密钥除外，目前不支持该密钥。

有关适用于 Amazon ECR 的存储库策略语句，请参阅Amazon ECR 存储库策略 (p. 14)。

检查用户是否具有所需权限

创建 IAM 策略之后，建议您检查它是否向用户授予了使用所需的特定 API 操作和资源的权限。请在策略投入生产之前进行检查。

首先，创建一个用于测试目的的 IAM 用户，然后将您创建的 IAM 策略与该测试用户关联起来。然后，以测试用户身份提出请求。您可以在控制台中提出测试请求，也可以使用 AWS CLI 提出测试请求。

> **Note**
>
> 您也可以使用 IAM 策略模拟器测试您的策略。有关策略模拟器的更多信息，请参阅 IAM 用户指南 中的使用 IAM 策略模拟器。

如果您测试的操作创建或修改了一种资源，您在提交请求时应该使用 `DryRun` 参数（或运行带有 `--dry-run` 选项的 AWS CLI 命令）。在这种情况下，调用会完成身份验证检查，但是不会完成该操作。例如，您可以检查用户能否终止特定实例，但不会真的终止它。如果测试用户具有所需的权限，请求会返回 `DryRunOperation`；否则，它会返回 `UnauthorizedOperation`。

如果策略未向用户授予您所期望的权限，或者策略过度宽松，可以根据需要调整策略。重新测试，直到获得预期的结果。

> **Important**
>
> 在其生效之前，它需要几分钟时间将策略更改为适合状态。因此，我们建议您在测试策略更新前，等候五分钟的时间。

如果身份验证检查失败，该请求将返回一个带有诊断信息的代码消息。您可以使用 `DecodeAuthorizationMessage` 操作对消息进行解码。有关更多信息，请参阅 AWS Security Token Service API Reference 中的 `DecodeAuthorizationMessage` 和 AWS CLI Command Reference 中的 decode-authorization-message。

Amazon ECR 托管策略

Amazon ECR 提供了一些托管策略，您可以将它们附加到 IAM 用户或 EC2 实例，以实现对 Amazon ECR 资源和 API 操作的不同级别的控制。您可以直接应用这些策略，或者也可以使用它们作为自行创建策略的起点。有关这些策略中提到的每个 API 操作的更多信息，请参阅 Amazon Elastic Container Registry API Reference 中的操作。

主题

- AmazonEC2ContainerRegistryFullAccess (p. 43)

- AmazonEC2ContainerRegistryPowerUser (p. 43)
- AmazonEC2ContainerRegistryReadOnly (p. 43)

AmazonEC2ContainerRegistryFullAccess

此策略授予对 Amazon ECR 的完全管理员访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:*"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonEC2ContainerRegistryPowerUser

此策略授予对 Amazon ECR 的高级用户访问权限，该权限允许对存储库进行读写访问，但不允许用户删除存储库或更改应用于存储库的策略文档。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "ecr:GetAuthorizationToken",
   "ecr:BatchCheckLayerAvailability",
   "ecr:GetDownloadUrlForLayer",
   "ecr:GetRepositoryPolicy",
   "ecr:DescribeRepositories",
   "ecr:ListImages",
   "ecr:DescribeImages",
   "ecr:BatchGetImage",
   "ecr:InitiateLayerUpload",
   "ecr:UploadLayerPart",
   "ecr:CompleteLayerUpload",
   "ecr:PutImage"
  ],
  "Resource": "*"
 }]
}
```

AmazonEC2ContainerRegistryReadOnly

此策略授予对 Amazon ECR 的只读访问权限，例如能够列出存储库和存储库中的映像，还能通过 Docker CLI 从 Amazon ECR 拉取映像。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
```

```
    "ecr:GetAuthorizationToken",
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:GetRepositoryPolicy",
    "ecr:DescribeRepositories",
    "ecr:ListImages",
    "ecr:DescribeImages",
    "ecr:BatchGetImage"
   ],
   "Resource": "*"
  }]
}
```

Amazon ECR API 操作支持的资源级权限

资源级权限 是指指定允许用户对哪些资源执行操作的能力。Amazon ECR 对资源级权限提供部分支持。这意味着，对于某些 Amazon ECR 操作，您可以控制何时允许用户使用这些操作。用户的访问可基于某些必须满足的条件或特定资源。

下表介绍当前支持资源级权限的 Amazon ECR API 操作，以及每个操作支持的资源和资源 ARN。

有关 Amazon ECR 操作的列表，请参阅 Amazon Elastic Container Registry API Reference 中的操作。

> **Important**
>
> 对于不支持资源级权限的 Amazon ECR API 操作，可以向用户授予使用该操作的权限。必须为策略语句的资源元素指定 * (星号) 通配符。

API 操作	资源存储库
BatchCheckLayerAvailability	arn:aws:ecr:*region*:*account*:repository/*my-repo*
BatchDeleteImage	arn:aws:ecr:*region*:*account*:repository/*my-repo*
BatchGetImage	arn:aws:ecr:*region*:*account*:repository/*my-repo*
CompleteLayerUpload	arn:aws:ecr:*region*:*account*:repository/*my-repo*
CreateRepository	此操作不支持资源级权限。
DeleteLifecyclePolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*
DeleteRepository	arn:aws:ecr:*region*:*account*:repository/*my-repo*
DeleteRepositoryPolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*
DescribeImages	arn:aws:ecr:*region*:*account*:repository/*my-repo*
DescribeRepositories	arn:aws:ecr:*region*:*account*:repository/*my-repo*
GetAuthorizationToken	此操作不支持资源级权限。
GetDownloadUrlForLayer	arn:aws:ecr:*region*:*account*:repository/*my-repo*
GetLifecyclePolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*
GetLifecyclePolicyPreview	arn:aws:ecr:*region*:*account*:repository/*my-repo*
GetRepositoryPolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*

API 操作	资源存储库
InitiateLayerUpload	arn:aws:ecr:*region*:*account*:repository/*my-repo*
ListImages	arn:aws:ecr:*region*:*account*:repository/*my-repo*
PutImage	arn:aws:ecr:*region*:*account*:repository/*my-repo*
PutLifecyclePolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*
SetRepositoryPolicy	arn:aws:ecr:*region*:*account*:repository/*my-repo*
StartLifecyclePolicyPreview	arn:aws:ecr:*region*:*account*:repository/*my-repo*
UploadLayerPart	arn:aws:ecr:*region*:*account*:repository/*my-repo*

使用基于标签的访问控制

利用 Amazon ECR CreateRepository API 操作，您可以在创建存储库时指定标签。有关更多信息，请参阅标记 Amazon ECR 存储库 (p. 18)。

要使用户能够在创建存储桶时标记存储桶，用户必须有权使用创建资源的操作（例如，`ecr:CreateRepository`）。如果在资源创建操作中指定了标签，则 Amazon 会对 `ecr:CreateRepository` 操作执行额外的授权，以验证用户是否具备创建标签的权限。

您可以通过 IAM 策略使用基于标签的访问控制。示例如下。

以下策略仅允许 IAM 用户创建存储库或将其标记为 `key=environment,value=dev`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateTaggedRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        },
        {
            "Sid": "AllowTagRepository",
            "Effect": "Allow",
            "Action": [
                "ecr:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": "dev"
                }
            }
        }
    ]
}
```

以下策略允许 IAM 用户访问所有存储库（除非这些存储库标记为 `key=environment,value=prod`）。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ecr:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ecr:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

创建 Amazon ECR IAM 策略

可以创建特定的 IAM 策略来限制您账户中的用户有权访问的调用和资源，然后将这些策略与 IAM 用户关联。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。有关 IAM 策略的更多一般信息，请参阅 IAM 用户指南 中的权限与策略。有关管理和创建自定义 IAM 策略的更多信息，请参阅管理 IAM 策略。

为用户创建 IAM 策略

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Policies 和 Create Policy。
3. 在 Create Policy (创建策略) 部分中，选择 Create Your Own Policy (创建您自己的策略) 旁边的 Select (选择)。
4. 对于 Policy Name (策略名称)，键入您自己的唯一名称，如 *AmazonECRUserPolicy*。
5. 对于 Policy Document，粘贴要应用于用户的策略。您可以将托管策略 (p. 42)作为起始点，创建自己的更严格或更宽松的 IAM 策略，以用于 Amazon ECR。
6. 选择 Create Policy。

将 IAM 策略附加到用户

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Users (用户)，然后选择要将策略附加到的用户。
3. 选择 Permissions、Add permissions。
4. 在 Grant permissions 部分，选择 Attach existing policies directly。
5. 选择您在上一个过程中创建的自定义策略，然后选择 Next: Review。
6. 检查详细信息，然后选择 Add permissions (添加权限)。

将 AWS CLI 与 Amazon ECR 结合使用

以下步骤可帮助您安装 AWS Command Line Interface 并登录 Amazon ECR。然后，您便可以创建映像存储库，将映像推送到该存储库中，并执行 Amazon ECR 中的其他常见场景。

AWS CLI 是用于管理 AWS 服务的统一工具。只通过一个工具进行下载和配置，您就可以使用命令行控制多个 AWS 服务并利用脚本来自动执行这些服务。有关更多信息，请参阅AWS Command Line Interface 用户指南。

有关可用于管理 AWS 资源的其他工具的更多信息，包括不同的 AWS 开发工具包、IDE 工具包和 Windows PowerShell 命令行工具，请参阅 http://www.amazonaws.cn/tools/。

主题

步骤 1：向您的默认注册表验证 Docker 身份

安装并配置 AWS CLI 后，向默认注册表验证 Docker CLI 的身份。这样一来，docker 命令可以通过 Amazon ECR 推送和拉取映像。AWS CLI 提供 get-login 命令来简化身份验证过程。

使用 get-login 对 Amazon ECR 注册表验证 Docker

Note

AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本 (17.06 或更高版本)，我们建议使用 1.11.91 或更高版本。您可以使用 aws --version 命令查看 AWS CLI 的版本。

1. 运行 aws ecr get-login 命令。以下示例适用于与创建请求的账户关联的默认注册表。要访问其他账户注册表，请使用 `--registry-ids` *`aws_account_id`* 选项。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。有关更多信息，请参阅 AWS CLI Command Reference 中的 get-login。

```
aws ecr get-login
```

输出：

```
docker login -u AWS -p password -e none https://aws_account_id.dkr.ecr.us-
east-1.amazonaws.com
```

Important

如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

结果输出是 docker login 命令，此命令可用于对 Amazon ECR 注册表验证 Docker 客户端。

2. 将 docker login 命令复制并粘贴到终端，授权您的 Docker CLI 访问注册表。此命令提供一个授权令牌，此令牌在 12 小时内对指定注册表有效。

   Note

   如果使用的是 Windows PowerShell，复制并粘贴这样的长字符串将不起作用。请使用以下命令。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。

   ```
   Invoke-Expression -Command (aws ecr get-login)
   ```

   Important

   在执行此 docker login 命令时，进程列表 (ps -e) 显示中将为系统上的其他用户显示命令字符串。由于 docker login 命令包含验证凭证，因此系统上的其他用户可按此方式查看凭证并使用这些凭证来获取对存储库的推送和拉取访问权会带来风险。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p` *`password`* 选项并在系统提示时输入密码来以交互方式登录。

步骤 2：获取 Docker 映像

首先必须有要推送的映像，才能将映像推送到 Amazon ECR。如果还没有可使用的映像，可以按照 Docker 基本知识 (p. 5)中的步骤创建一个。或者，从 Docker Hub 中拉取要包含在 Amazon ECR 注册表中的映像。要将 `ubuntu:trusty` 映像从 Docker Hub 拉取到本地系统，可运行以下命令：

```
$ docker pull ubuntu:trusty
trusty: Pulling from library/ubuntu
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:3cb273da02362a6e667b54f6cf907edd5255c706f9de279c97cfccc7c6988124
Status: Downloaded newer image for ubuntu:trusty
```

步骤 3：创建存储库

现在您已拥有可推送到 Amazon ECR 的映像，还必须创建一个存储库来保存它。在本示例中，您创建一个名称为 `ubuntu` 的存储库，稍后将推送 `ubuntu:trusty` 映像到这里。要创建存储库，请运行以下命令：

```
$ aws ecr create-repository --repository-name ubuntu
{
    "repository": {
        "registryId": "111122223333",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-east-1:111122223333:repository/ubuntu"
    }
}
```

步骤 4：推送映像到 Amazon ECR

现在您可以推送映像到上一部分中创建的 Amazon ECR 存储库。您使用 docker CLI 推送映像，但必须满足一些先决条件才能正常工作：

- 安装最低版本的 docker：1.7
- 已使用 docker login 配置 Amazon ECR 授权令牌。
- Amazon ECR 存储库存在且用户有向该存储库推送的权限。

在满足这些先决条件后，即可将映像推送到您在帐户的默认注册表中新创建的存储库中。

标记映像并推送到 Amazon ECR

1. 列出您存储在本地的映像，以识别要标记和推送的映像。

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
 SIZE
ubuntu              trusty              e9ae3c220b23        3 weeks ago         187.9
 MB
```

2. 标记映像并推送到存储库。

```
$ docker tag ubuntu:trusty aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu:trusty
```

3. 推送映像。

```
$ docker push aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu:trusty
The push refers to a repository [aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu]
 (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
trusty: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b
 size: 6774
```

步骤 5：从 Amazon ECR 拉取映像

在推送映像到 Amazon ECR 存储库后，可以从其他位置拉取该映像。可使用 docker CLI 拉取映像，但必须满足以下几个先决条件才能正常使用：

- 安装最低版本的 docker：1.7
- 已使用 docker login 配置 Amazon ECR 授权令牌。
- Amazon ECR 存储库存在且用户有从该存储库拉取的权限。

在满足这些先决条件后，即可拉取您的映像。要从 Amazon ECR 拉取示例映像，请运行以下命令：

```
$ docker pull aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu:trusty
trusty: Pulling from ubuntu
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
```

```
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b
Status: Downloaded newer image for aws_account_id.dkr.ecr.us-east-1.amazonaws.com/
ubuntu:trusty
```

步骤 6：删除映像

如果您不再需要一个存储库中的某个映像，则可以使用 batch-delete-image 命令将其删除。要删除映像，您必须指定它所在的存储库，并指定映像的 `imageTag` 或 `imageDigest` 值。以下示例删除 `ubuntu` 存储库中映像标签为 `trusty` 的映像。

```
$ aws ecr batch-delete-image --repository-name ubuntu --image-ids imageTag=trusty
{
    "failures": [],
    "imageIds": [
        {
            "imageTag": "trusty",
            "imageDigest":
 "sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b"
        }
    ]
}
```

步骤 7：删除存储库

如果您不再需要整个存储库中的所有映像，您可以删除存储库。默认情况下，您不能删除包含映像的存储库；但是，`--force` 标记允许此操作。要删除包含映像的存储库 (及其中的所有映像)，请运行以下命令：

```
 $ aws ecr delete-repository --repository-name ubuntu --force
{
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-east-1:aws_account_id:repository/ubuntu",
        "createdAt": 1457671643.0,
        "repositoryUri": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu"
    }
}
```

Amazon ECR 服务限制

下表提供了 Amazon Elastic Container Registry (Amazon ECR) 对于 AWS 账户的默认限制，这些限制可以更改。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务限制。

资源	默认限制
每个区域的最大存储库数	1,000
每个存储库的最大图像数	1000
每秒、每区域、每账户的 `GetAuthorizationToken` API 事务数	20 个持续事务，可以突增至 200 个 *
每秒、每区域、每账户的针对存储库的 `docker pull` 事务数	200 个持续事务，可以突增至 400 个 *
每秒、每区域、每账户的针对存储库的 `docker pull layer` 事务数	200 个持续事务，可以突增至 400 个 *
每秒、每区域、每账户的针对存储库的 `docker push` 事务数	10 个持续事务，可以突增至 40 个 *

* 在每个区域中，每个账户会收到一个最多可存储特定数量的积分的存储桶，具体取决于事务。这些积分以指定的每秒维持率补充。例如，对于 `GetAuthorizationToken` API 事务，存储桶最多可存储 200 点积分。您可以实现每秒 200 个 `GetAuthorizationToken` API 事务（持续一秒），然后无限期地维持每秒 20 个事务。

下表提供了 Amazon ECR 和 Docker 映像的其他限制，这些限制无法更改。

> **Note**
>
> 下表中提到的层分段信息仅适用于直接调用 Amazon ECR API 操作，以便为映像推送操作启动分段上传的客户。估计这种情况很罕见。
> 我们建议您使用 docker CLI 来拉取、标记和推送映像。

资源	默认限制
每个映像层数上限	127 (此为当前的 Docker 限制)
每个映像的最大标签数	100
层大小上限 **	10,000 MiB
层分段大小上限	10 MiB
层分段大小下限	5 MiB (上传中的最后一个层分段例外)
层分段数上限	1000
生命周期策略中的规则的最大数量	50
生命周期策略的最大长度	30720

** 此处所列的层大小上限由层分段大小上限 (10 MiB) 乘以层分段数上限 (1000) 计算得出。

Amazon ECR 使用率报告

AWS 提供了称为 Cost Explorer 的免费报告工具，该工具可让您分析 Amazon ECR 资源的成本和使用率。

Cost Explorer 是一款免费工具，可用于查看使用率和成本的图表。您可以查看过去 13 个月的数据，并预测您在接下来三个月内可能产生的费用。您可以使用 Cost Explorer 查看有关您一段时间内在 AWS 资源方面的费用的模式、确定需要进一步查询的方面以及查看可用于了解您的成本的趋势。您还可以指定数据的时间范围，并按天或按月查看时间数据。

成本和使用率报告中的计量数据显示跨所有 Amazon ECR 存储库的使用率。该计量数据包括每个存储库的 。存储库将不会有关联的成本。您可以使用元数据标签来标识存储库。有关更多信息，请参阅 标记资源以便于计费 (p. 19)。

有关创建 AWS 成本和使用率报告的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的 AWS 成本和使用率报告。

使用 Amazon ECR 记录 AWS CloudTrail API 调用

Amazon ECR 与 AWS CloudTrail 集成，后者是在 Amazon ECR 中提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将对 Amazon ECR 的所有 API 调用作为事件捕获，包括来自 Amazon ECR 控制台的调用和对 Amazon ECR API 的代码调用。如果您创建跟踪，则可以使 CloudTrail 事件持续交付到 Amazon S3 存储桶（包括 Amazon ECR 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台的 Event history (事件历史记录) 中查看最新事件。通过使用此信息，可以确定向 Amazon ECR 发出的请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

有关更多信息，请参阅 AWS CloudTrail User Guide。

CloudTrail 中的 Amazon ECR 信息

在您创建 CloudTrail 账户时，即针对该账户启用了 AWS。Amazon ECR 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 Amazon ECR 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个区域的 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Amazon ECR 操作均由 CloudTrail 记录下来并记载到 Amazon Elastic Container Registry API Reference 中。例如，对 `GetAuthorizationToken`、`CreateRepository` 和 `SetRepositoryPolicy` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail `userIdentity` 元素。

了解 Amazon ECR 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

Note

为提高可读性，这些示例已进行格式化处理。在 CloudTrail 日志文件，所有条目和事件都连接成一行。此外，该示例限于一个 Amazon ECR 条目。在实际的 CloudTrail 日志文件中，可以看到来自多个 AWS 服务的条目和事件。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateRepository` 操作。

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-11T21:54:07Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            }
        }
    },
    "eventTime": "2018-07-11T22:17:43Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "CreateRepository",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "repositoryName": "testrepo"
    },
    "responseElements": {
        "repository": {
            "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "repositoryName": "testrepo",
            "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
            "createdAt": "Jul 11, 2018 10:17:44 PM",
            "registryId": "123456789012"
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
            "accountId": "123456789012"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Amazon ECR 问题排查

本章帮助您查找 Amazon Elastic Container Registry (Amazon ECR) 的诊断信息，并为常见问题和错误消息提供故障排除步骤。

主题

启用 Docker 调试输出

要开始调试任何 Docker 相关问题，都需要首先在您的主机实例上运行的 Docker 守护程序中启用 Docker 调试输出。如果在 Amazon ECS 容器实例中使用从 Amazon ECR 拉取的映像，有关启用 Docker 调试的更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的启用 Docker 调试输出。

启用 AWS CloudTrail

有关 Amazon ECR 所返回错误的其他信息，可以通过启用 AWS CloudTrail 进行查找，它是为 AWS 账户记录 AWS 调用的一项服务。CloudTrail 可将日志文件传输给 Amazon S3 存储桶。通过使用 CloudTrail 收集的信息，您可以确定向 AWS 服务成功发出了哪些请求、发出请求的用户、请求时间等等。要了解有关 CloudTrail 的更多信息（包括如何启用该服务及如何查找日志文件），请参阅 AWS CloudTrail User Guide。有关将 CloudTrail 与 Amazon ECR 配合使用的更多信息，请参阅使用 Amazon ECR 记录 AWS CloudTrail API 调用 (p. 54)。

为 Amazon ECR 优化性能

以下部分提供了在使用 Amazon ECR 时可用于优化性能的设置和策略建议。

使用 Docker 1.10 及以上版本可利用同时层上传

Docker 映像由层组成，是映像的中间构建阶段。Dockerfile 的每一行都会创建新层。当使用 Docker 1.10 及以上版本时，Docker 在默认情况下会在上传至 Amazon ECR 的同时推送尽可能多的层，从而缩短上传时间。

使用较小基本映像

通过 Docker Hub 提供的默认映像，可能包含您的应用程序不需要的很多依赖项。请考虑使用其他人在 Docker 社区创建并维护的较小映像，或使用 Docker 最小映像构建您自己的基本映像。有关更多信息，请参阅 Docker 文档中的创建基本映像。

更早将更改最少的依赖性放入您的 Dockerfile

Docker 缓存层，可加速构建时间。如果从最后一次构建至今，某一层上没有任何更改，则 Docker 将使用缓存版本，而不重新构建层。但是，每层都依赖之前出现的层。如果层发生更改，则 Docker 不仅重新编译该层，也会重新编译该层之后出现的所有层。

为了尽量缩短重新构建 Dockerfile 并重新上传层所需的时间，可考虑早些时候将更改频率最低的依赖项放入 Dockerfile。将经常更改的依赖项（如应用程序的源代码）稍后放入堆栈。

链接命令以避免不必要文件的存储

在层中创建的中间文件会作为该层的一部分保留，即使该层在后续层中被删除。考虑以下示例：

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

在本示例中，第一个和第二个 RUN 命令创建的层包含原始 .tar.gz 文件及其所有解压内容。即使第四个 RUN 命令已删除 .tar.gz 文件。这些命令可以链接在一起，构成单独的运行语句，以确保最终 Docker 映像中不包含不必要的文件。

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

使用最近的区域终端节点

通过确保使用最靠近所运行应用程序的区域终端节点，可以减少从 Amazon ECR 拉取映像的延迟。如果应用程序在 Amazon EC2 实例上运行，可以使用以下 shell 代码从实例的可用区获取区域：

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone |\
     sed -n 's/\(\d*\)[a-zA-Z]*$/\1/p')
```

可以使用 --region 参数将该区域传递给 AWS CLI 命令，或使用 aws configure 命令将该区域设为某个配置文件的默认区域。您还可以在使用 AWS 软件开发工具包进行调用时设置区域。有关更多信息，请参阅适用于特定编程语言的软件开发工具包文档。

使用 Amazon ECR 时通过 Docker 命令纠正错误

主题

有时，针对 Amazon ECR 运行 Docker 命令可能导致错误消息。一些常见错误消息和可能的解决办法解释如下。

从 Amazon ECR 存储库拉取映像时，出现错误：“Filesystem Verification Failed”(文件系统验证失败)或“404: Image Not Found”(404：找不到映像)

在 Docker 1.9 或更高版本中使用 docker pull 命令从 Amazon ECR 存储库拉取映像时，可能会收到错误 `Filesystem verification failed`。如果使用的是 1.9 之前的 Docker 版本，则可能收到错误 `404: Image not found`。

以下为一些可能的原因及它们的解释。

本地磁盘已满

如果运行 docker pull 命令的本地磁盘已满，那么对本地文件计算的 SHA-1 哈希值可能与 Amazon ECR 计算的 SHA-1 哈希值不同。确保本地磁盘有足够的剩余空间可存储所拉取的 Docker 映像。为腾出空间存储新映像，可以删除旧映像。使用 docker images 命令可查看所有已下载到本地的 Docker 映像的列表及这些映像的大小。

由于网络错误，客户端无法连接到远程存储库

调用 Amazon ECR 存储库需要 Internet 连接正常。验证网络设置，然后验证其他工具和应用程序是否可以访问 Internet 上的资源。如果在私有子网中对 Amazon EC2 实例运行 docker pull，请验证该子网是否具有连接至 Internet 的路由。可使用网络地址转换 (NAT) 服务器或托管的 NAT 网关。

目前，调用 Amazon ECR 存储库还要求通过您的公司防火墙访问 Amazon Simple Storage Service (Amazon S3)。如果贵企业或组织使用的是允许服务终端节点的防火墙软件或 NAT 设备，请确保当前区域的 Amazon S3 服务终端节点在允许范围内。

如果您通过 HTTP 代理使用 Docker，可以对 Docker 进行相应的代理设置。有关更多信息，请参阅 Docker 文档中的 HTTP 代理。

从 Amazon ECR 拉取映像时，出现错误：“Filesystem Layer Verification Failed”(文件系统分层验证失败)

您可能在使用 docker pull 命令拉取映像时收到错误 `image image-name not found`。如果检查 Docker 日志，可能会看到与下面类似的错误：

```
filesystem layer verification failed for digest sha256:2b96f...
```

此错误表示映像的一个或多个层下载失败。以下为一些可能的原因及它们的解释。

您正在使用旧版本的 Docker

在使用低于 1.10 的 Docker 版本时，有少数情况会出现此错误。请将您的 Docker 客户端升级至 1.10 或更高版本。

您的客户端遇到网络错误或磁盘错误

如前文对 `Filesystem verification failed` 消息的讨论中所述，磁盘已满或网络问题可能会导致一个或多个层无法下载。请遵循上述建议确保您的文件系统未满，并且您在网络中有对 Amazon S3 的访问权限。

推送到存储库时出现 HTTP 403 错误或“no basic auth credentials”(没有基础级验证凭证) 错误

有时，即使您已使用 aws ecr get-login 命令成功通过 Docker 身份验证，也可能会从 docker push 命令收到“HTTP 403 (Forbidden) (HTTP 403 (禁止访问))”错误或者错误消息 `no basic auth credentials`。以下是此问题的一些已知的原因：

您已验证到其他区域

身份验证请求与特定的区域相关联，不能跨区域使用。例如，如果您从 美国西部（俄勒冈） 获得授权令牌，不能使用它对您在 美国东部（弗吉尼亚北部） 的存储库进行身份验证。要解决该问题，请确保在身份验证和 docker push 命令调用时使用相同的区域。

您已进行身份验证以推送到错误 AWS 账户上的存储库

如果使用的是来自一个 AWS 账户的 IAM 用户，但尝试推送到另一个账户中托管的存储库，则必须在调用 aws ecr get-login 时显式指定 --registry-ids 参数。否则，您获得的 Docker 登录命令在默认情况下，将只授权您推送到托管 IAM 用户的同一账户上的存储库。您无权推送到托管存储库的账户。始终确保来自 aws ecr get-login 的响应的存储库 URL 与您要推送到的存储库 URL 匹配，包括 URL 的账户 ID 部分。

您的令牌已过期。

使用 `GetAuthorizationToken` 操作获取的令牌，默认令牌有效期为 12 小时。但是，如果使用临时安全凭证机制来验证身份和接收令牌，则令牌的有效期与临时凭证的持续时间是一致的。临时安全凭证机制包括多重身份验证 (MFA) 或 AWS Security Token Service。例如，如果通过代入角色来调用 aws ecr get-login 命令，则授权令牌将在 15 分钟到 1 个小时后过期。具体取决于调用 aws sts assume-role 命令时所用的设置。

`wincred` 凭证管理器中的错误

某些版本的适用于 Windows 的 Docker 使用名为 `wincred` 的凭证管理器，但它无法正确处理由 aws ecr get-login 生成的 Docker 登录命令（有关更多信息，请参阅 https://github.com/docker/docker/issues/22910）。可以运行作为输出的 Docker 登录命令，但如果尝试推送或拉取镜像，这些命令会失败。修复这个错误的方法是，对于从 aws ecr get-login 输出的 Docker 登录命令，删除其注册表参数中的 `https://` 方案。如下所示为不带 HTTPS 方案的 Docker 登录命令示例。

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

Amazon ECR 错误消息问题排查

主题

有时，通过 Amazon ECS 控制台或 AWS CLI 或触发的 API 调用会存在错误消息。一些常见错误消息和可能的解决办法解释如下。

运行 aws ecr get-login 时出现错误：“Error Response from Daemon: Invalid Registry Endpoint”(守护程序响应出错：注册表终端节点无效)

运行 aws ecr get-login 命令获取 Amazon ECR 存储库的登录凭证时，您可能看到以下错误：

```
Error response from daemon: invalid registry endpoint
    https://xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/v0/: unable to ping registry
 endpoint
    https://xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/v0/
v2 ping attempt failed with error: Get https://xxxxxxxxxxxx.dkr.ecr.us-
east-1.amazonaws.com/v2/:
    dial tcp: lookup xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com on 172.20.10.1:53:
    read udp 172.20.10.1:53: i/o timeout
```

运行 Docker 工具箱、Docker for Windows 或 Docker for Mac 的 MacOS x 和 Windows 系统，可能出现此错误。导致此问题的原因通常是：其他应用程序在通过本地网关更改路由器 (192.168.0.1) 时，虚拟机必须通过调用本地网关才能访问 Amazon ECR 服务。如果使用 Docker 工具箱时出现此错误，通常可以通过重启 Docker 系统环境，或重新启动本地客户端的操作系统来解决。如果该方法未能解决问题，可使用 docker-machine ssh 命令登录容器实例。可在外部主机上执行 DNS 查找，以验证其结果是否与本地主机上的结果相同。如果结果不同，请参考 Docker 工具箱的文档，确保 Docker 系统环境已正确配置。

HTTP 429：请求过多或 ThrottleException

您可能会从一个或多个 Amazon ECR 命令或 API 调用收到 `429: Too Many Requests` 错误或 `ThrottleException` 错误。如果将 Docker 工具用于 Amazon ECR，那么对于 Docker 1.12.0 版及更高版本，可能会显示错误消息 `TOOMANYREQUESTS: Rate exceeded`。对于 1.12.0 以下的 Docker 版本，您可能会看到错误 `Unknown: Rate exceeded`。

这表示由于您在短时间内重复调用 Amazon ECR 中的单个终端节点，您的请求已受限制。单个用户在一段时间内，调用单个终端节点的次数超过特定阈值时，就会产生限制。

Amazon ECR 中不同的 API 操作有不同的限制。

例如，`GetAuthorizationToken` 操作的限制为每秒 20 个事务 (TPS)，允许高达 200 TPS 的突增。在每个区域，每个账户会收到一个可存储多达 200 点 `GetAuthorizationToken` 积分的存储桶。这些积分以每秒 20 点的速度补充。如果您的存储桶有 200 点积分，则可实现每秒 200 个 `GetAuthorizationToken` API 事务 (持续一秒)，然后无限期地维持每秒 20 个事务。

要处理限制错误，请在代码中实施增量退避重试函数。有关更多信息，请参阅 Amazon Web Services 一般参考中的 AWS 中的错误重试和指数退避。

HTTP 403：“User [arn] is not authorized to perform [operation]”(用户 [arn] 没有执行 [operation] 的权限)

尝试通过 Amazon ECR 执行操作时，您可能会收到以下错误：

```
$ aws ecr get-login
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken
 operation:
    User: arn:aws:iam::account-number:user/username is not authorized to perform:
    ecr:GetAuthorizationToken on resource: *
```

这表示您的用户没有获得使用 Amazon ECR 的权限，或者这些权限设置不正确。尤其是在对 Amazon ECR 执行操作时，请验证是否已授予用户访问该存储库的权限。有关创建和验证 Amazon ECR 权限的更多信息，请参阅 Amazon ECR IAM 策略和角色 (p. 39)。

HTTP 404：“Repository Does Not Exist”(存储库不存在) 错误

如果您指定了当前不存在的 Docker Hub 存储库，Docker Hub 会自动创建存储库。但在使用 Amazon ECR 时，新存储库必须在使用前显式创建。这会防止意外创建新存储库 (例如，由于输入错误)，也可确保为所有新存储库明确分配适当的安全访问策略。有关创建存储库的更多信息，请参阅Amazon ECR 存储库 (p. 12)。

文档历史记录

下表列出了自 Amazon ECR 上一次发布以来对文档所做的重要更改。我们还经常更新文档来处理您发送给我们的反馈意见。

- 最近文档更新时间：2017 年 11 月 21 日

变更	描述	日期
为资源添加标签	Amazon ECR 增加了对为存储库添加元数据标签的支持。 有关更多信息，请参阅标记 Amazon ECR 存储库 (p. 18)。	2018 年 12 月 18 日
Amazon ECR 名称变更	Amazon Elastic Container Registry 已更名（原来称为 Amazon EC2 Container Registry）。	2017 年 11 月 21 日
生命周期策略	Amazon ECR 生命周期策略使您能够指定存储库中映像的生命周期管理。 有关更多信息，请参阅 Amazon ECR 生命周期策略 (p. 28)。	2017 年 10 月 11 日
Amazon ECR 支持 Docker Image Manifest 2、Schema 2	Amazon ECR 现已支持 Docker Image Manifest V2 Schema 2 (与 Docker 版本 1.10 和更高版本配合使用) 有关更多信息，请参阅 容器映像清单格式 (p. 25)。	2017 年 1 月 27 日
Amazon ECR 正式发布	Amazon Elastic Container Registry (Amazon ECR) 是一项托管 AWS Docker 注册表服务，它安全、可扩展且可靠。	2015 年 12 月 21 日

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 AWS 词汇表。