Amazon ECR Public

User Guide

API Version 2015-09-21

Amazon ECR Public: User Guide

Table of Contents

What Is Amazon Elastic Container Registry Public?

Amazon Elastic Container Registry Public is a managed AWS container image registry service that is secure, scalable, and reliable. Amazon ECR supports public image repositories with resource-based permissions using AWS IAM so that specific users can access your public repositories to push images. Developers can use their preferred CLI to push and manage Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts. Your images are publicly available to pull, either anonymously or using an Amazon ECR Public authentication token.

> **Note**
> Amazon ECR supports private container image repositories as well. For more information, see What is Amazon ECR in the *Amazon Elastic Container Registry User Guide*.

The AWS container services team maintains a public roadmap on GitHub. It contains information about what the teams are working on and allows all AWS customers the ability to give direct feedback. For more information, see AWS Containers Roadmap.

Components of Amazon ECR Public

Amazon ECR Public contains the following components:

Amazon ECR Public Gallery

The Amazon ECR Public Gallery is the public portal that lists all public repositories hosted on Amazon ECR Public. Visit the Amazon ECR Public Gallery at https://gallery.ecr.aws. For more information, see Using the Amazon ECR Public Gallery (p. 11).

Registry

A public registry is provided to each AWS account; you can create public image repositories in your public registry and store images in them. For more information, see Amazon ECR public registries (p. 12).

Authorization token

Your client must authenticate to a public registry as an AWS user before it can push images to a public repository. For image pulls, Amazon ECR Public accepts both anonymous pulls and pulls using an authentication token. For more information, see Registry authentication (p. 13).

Repository

An Amazon ECR image repository contains your Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts. For more information, see Amazon ECR public repositories (p. 17).

Repository policy

You can control access to your repositories and the images within them with repository policies. For more information, see Public repository policies (p. 27).

Image

You can push and pull container images to your repositories. You can use these images locally on your development system, or you can use them in Amazon ECS task definitions and Amazon EKS pod specifications.

Setting up with Amazon ECR

If you've signed up for AWS and have been using Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS), you are close to being able to use Amazon ECR. The setup process for those two services is similar, as Amazon ECR is an extension of both services. When using the AWS CLI with Amazon ECR, we recommend that you use a version of the AWS CLI that supports the latest Amazon ECR features. If you do not see support for an Amazon ECR feature in the AWS CLI, you should upgrade to the latest version. For more information, see http://aws.amazon.com/cli/.

Complete the following tasks to get set up to push a container image to Amazon ECR for the first time. If you have already completed any of these steps, you may skip them and move on to the next step.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.
2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

- For your daily administrative tasks, grant administrative access to an administrative user in AWS IAM Identity Center (successor to AWS Single Sign-On).

  For instructions, see Getting started in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Getting started with Amazon ECR Public

Get started with Amazon ECR public repositories by creating your first public repository and setting your public registry settings in the Amazon ECR console. The Amazon ECR console guides you through the process to get started.

Prerequisites

Before you begin, be sure that you've completed the steps in Setting up with Amazon ECR (p. 2).

To build, tag, and push container images to your Amazon ECR public repositories you must have the AWS CLI and a CLI client, for example the Docker CLI, to do this. Docker is a common tool for building container images. Docker is available on many different operating systems, including most modern Linux distributions, like Ubuntu, and even Mac OSX and Windows. You don't need a local development system to use Docker. If you are using Amazon EC2 already, you can launch an Amazon EC2 instance and install Docker to get started. For more information about how to install Docker on your particular operating system, go to the Docker installation guide.

To use the AWS CLI with Amazon ECR Public, install the latest AWS CLI version. For information about installing the AWS CLI or upgrading to the latest version, see Installing the AWS CLI version 2 in the *AWS Command Line Interface User Guide*.

Create a public repository

To create a public image repository (AWS Management Console)

A repository is where you store your Docker or Open Container Initiative (OCI) images in Amazon ECR that you want to make publicly available for others to pull.

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/get-started.
2. Choose **Get Started**.
3. For **Visibility settings**, choose **Public**.
4. For **Repository name**, enter a unique name for your public repository.
5. (Optional) For **Repository logo**, choose **Upload file** and select a local image file to use as the repository logo.

   Note
   The repository logo is only publicly visible in the Amazon ECR Public Gallery for verified accounts. A verified account is an account that is AWS Marketplace certified.
6. For **Short description** enter a description of the repository. The description field is displayed on the Amazon ECR Public Gallery in the search results and on the repository detail page.
7. For **Content types** select the operating system and system architecture tags to associate with the repository. These tags are publicly displayed in the Amazon ECR Public Gallery as badges on the repository and are used as search filters.
8. For **About**, enter a detailed description for the repository. This text should be in Github Flavored Markdown format. For format examples, see Repository catalog data (p. 19). This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page.

9. For **Usage**, enter details about how to use the images in the repository. This text should be in Github Flavored Markdown format. For format examples, see Repository catalog data (p. 19). This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page.
10. Choose **Create repository**.

Push a Docker image

Build, tag, and push a Docker image

You can build, tag, and push a container image using the Docker CLI. You can use a container image that you have built from a Dockerfile or one that you pulled from another registry, such as a private Amazon ECR repository or Docker Hub and then push the tagged image to your public repository. For more detailed steps on using the Docker CLI, see .

1. Select the public repository you created and choose **View push commands** to view the steps to push an image to your new repository.
2. Run the login command that authenticates your Docker client to your registry by pasting the command from the console into a terminal window. This command provides an authorization token that is valid for 12 hours.
3. (Optional) If you have a Dockerfile for the image to push, build the image and tag it for your new repository. Pasting the **docker build** command from the console into a terminal window. Make sure that you are in the same directory as your Dockerfile.
4. Tag the image with the URI of your public registry and your new repository by pasting the **docker tag** command from the console into a terminal window. The console command assumes that your image was built from a Dockerfile in the previous step. If you did not build your image from a Dockerfile, replace the first instance of *`repository`*`:latest` with the image ID or image name of your local image to push.
5. Push the newly tagged image to your repository by pasting the **docker push** command into a terminal window.
6. Choose **Close**.

Edit the registry settings

(Optional) To edit your public registry settings

Each AWS account is provided with a default public Amazon ECR registry. The public registry is assigned a default alias after you have created your first public repository or requested a custom alias. Once Amazon ECR approves your custom alias request, it will appear when describing your registry as well as on your public repositories on the Amazon ECR Public Gallery.

The registry alias is part of the repository URI that is used to pull the images in the public repository. The following steps walk you through requesting a custom alias and setting a display name for your registry.

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. From the navigation bar, choose the Region to edit your public registry settings in.
3. In the navigation pane, choose **Registries**.
4. On the **Registries** page, select your **Public** registry and then choose **Edit**.
5. For **Custom alias**, enter a custom alias to request.
6. For **Display name**, enter a display name for your registry.
7. Choose **Save changes**.

Quick start: Publishing to Amazon ECR Public using the AWS CLI

This quick start guide walks you through the steps needed to create a Docker image, publish the image to a public repository, pull the image down from the Amazon ECR Public Gallery, and then clean up the resources using the Docker CLI and the AWS CLI.

For more information on the other tools available for managing your AWS resources, including the different AWS SDKs, IDE toolkits, and the Windows PowerShell command line tools, see http://aws.amazon.com/tools/.

Prerequisites

Before you begin, be sure that you've completed the steps in Setting up with Amazon ECR (p. 2).

To build, tag, and push container images to your Amazon ECR public repositories you must have the AWS CLI and a container CLI client, for example Docker. Docker is a common tool for building container images. For more information about how to install Docker, see Docker installation guide.

To use the AWS CLI with Amazon ECR Public, install the latest AWS CLI version. For information, see Installing the AWS CLI version 2 in the *AWS Command Line Interface User Guide*.

Step 1: Create a Docker image

In this step, you create a Docker image of a simple web application, and test it on your local system or Amazon EC2 instance.

To create a Docker image of a simple web application

1. Create a file called `Dockerfile`. A Dockerfile is a manifest that describes the base image to use for your Docker image and what you want installed and running on it. For more information about Dockerfiles, go to the Dockerfile Reference.

```
touch Dockerfile
```

2. Edit the `Dockerfile` you just created and add the following content.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html
```

```
# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

This Dockerfile uses the public Amazon Linux 2 image hosted on Amazon ECR Public. The `RUN` instructions update the package caches, installs some software packages for the web server, and then write the "Hello World!" content to the web server's document root. The `EXPOSE` instruction exposes port 80 on the container, and the `CMD` instruction starts the web server.

3. Build the Docker image from your Dockerfile.

   Note
   Some versions of Docker may require the full path to your Dockerfile in the following command, instead of the relative path shown below.

```
docker build -t hello-world .
```

4. List your container image.

```
docker images --filter reference=hello-world
```

   Output:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago       194MB
```

5. Run the newly built image. The `-p 80:80` option maps the exposed port 80 on the container to port 80 on the host system. For more information about **docker run**, go to the Docker run reference.

```
docker run -t -i -p 80:80 hello-world
```

   Note
   Output from the Apache web server is displayed in the terminal window. You can ignore the `"Could not reliably determine the server's fully qualified domain name"` message.

6. Open a browser and point to the server that is running Docker and hosting your container.

   - If you are using an EC2 instance, this is the **Public DNS** value for the server, which is the same address you use to connect to the instance with SSH. Make sure that the security group for your instance allows inbound traffic on port 80.
   - If you are running Docker locally, point your browser to http://localhost/.
   - If you are using **docker-machine** on a Windows or Mac computer, find the IP address of the VirtualBox VM that is hosting Docker with the **docker-machine ip** command, substituting *`machine-name`* with the name of the docker machine you are using.

```
docker-machine ip machine-name
```

   You should see a web page with your "Hello World!" statement.

7. Stop the Docker container by typing **Ctrl + c**.

Step 2: Authenticate to a public registry

After you have installed and configured the AWS CLI, authenticate the Docker CLI to your public registry. That way, the **docker** command can push to and pull images from an Amazon ECR public repository. The AWS CLI provides a **get-login-password** command to simplify the authentication process.

To authenticate Docker to an Amazon ECR public registry with get-login-password, run the **aws ecr-public get-login-password --region us-east-1** command. The Amazon ECR Public registry requires authentication in the `us-east-1` Region, so you need to specify `--region us-east-1` each time you authenticate. The authentication token received gives you access to each public registry your IAM principal has access to. When passing the authentication token to the **docker login** command, use the value `AWS` for the username and specify `public.ecr.aws`, which is the common public registry URI.

Important
If you receive an error, install or upgrade to the latest version of the AWS CLI. For more information, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --
password-stdin public.ecr.aws
```

Step 3: Create a public repository

Now that you have an image to push to Amazon ECR Public, you can create a public repository. In this example, you create a public repository called `ecr-tutorial` to which you later push the `hello-world:latest` image. All public repositories that contain an image are publicly visible in the Amazon ECR Public Gallery so we will specify some catalog data for the repository.

Create a file named `repositorycatalogdata.json` with the following contents. For this tutorial we are going to include a repository logo, which is named `myrepoimage.png` and is in the same directory as the `repositorycatalogdata.json` file we are creating.

Note
When creating a repository logo, the supported image dimensions for both height and width should be a minimum of 60 pixels and a maximum of 2048 pixels. The maximum logo file size is 500 KB.

```
{
    "description": "This is a test repo for an Amazon ECR tutorial.",
    "architectures": [
        "x86"
    ],
    "operatingSystems": [
        "Linux"
    ],
    "logoImageBlob": "$(cat myrepoimage.png |base64 -w 0)",
    "aboutText": "This repository is used as a tutorial only.",
    "usageText": "This repository is not for public use."
}
```

Use the catalog data file we just created to run the following `create-repository` command. Your repository URI is included in the response, which you will need in the next step for pushing an image to the repository.

```
aws ecr-public create-repository \
     --repository-name ecr-tutorial \
     --catalog-data file://repositorycatalogdata.json \
```

```
    --region us-east-1
```

Step 4: Push an image to Amazon ECR Public

Now you can push your image to the Amazon ECR public repository you created in the previous section. You use the Docker CLI to push images, but there are a few prerequisites that must be satisfied for this to work properly:

- The Amazon ECR Public authorization token has been configured with **docker login**.
- The Amazon ECR public repository exists and the user has access to push to the repository.

After those prerequisites are met, you can push your image to your newly created repository in the default registry for your account.

To tag and push an image to Amazon ECR Public

1. List the images you have stored locally to identify the image to tag and push.

   ```
   docker images
   ```

   Output:

   ```
   REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
    SIZE
   hello-world         latest              e9ffedc8c286        4 minutes ago       241MB
   ```

2. Tag the image to push to your repository with your public repository URI which was in the response to the `create-repository` call you made in the previous step.

   ```
   docker tag hello-world:latest public.ecr.aws/registry_alias/ecr-tutorial
   ```

3. Push the image.

   ```
   docker push public.ecr.aws/registry_alias/ecr-tutorial
   ```

   Output:

   ```
   The push refers to a repository [public.ecr.aws/registry_alias/ecr-tutorial] (len: 1)
   e9ae3c220b23: Pushed
   a6785352b25c: Pushed
   0998bf8fb9e9: Pushed
   0a85502c06c9: Pushed
   latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95EXAMPLE
    size: 1569
   ```

Step 5: Pull an image from the Amazon ECR Public Gallery

After your image has been pushed to your Amazon ECR public repository, you can pull it from other locations. It is considered best practice to authenticate prior to pulling images from the public gallery. If you need to reauthenticate, see Step 2: Authenticate to a public registry (p. 8).

Note
Unauthenticated pulls are allowed, but have a lower rate limit than authenticated pulls. For more information, see Amazon ECR Public service quotas (p. 64).

View your repository on the Amazon ECR Public Gallery.

```
https://gallery.ecr.aws/registry_alias/ecr-tutorial
```

```
docker pull public.ecr.aws/registry_alias/ecr-tutorial/hello-world:latest
```

Step 6: Delete a public image

If you decide that you no longer need or want an image in one of your repositories, you can delete it with the **batch-delete-image** command. To delete an image, you must specify the repository that it is in and either a `imageTag` or `imageDigest` value for the image. The example below deletes an image in the `hello-world` repository with the image tag `latest`.

```
aws ecr-public batch-delete-image \
      --repository-name ecr-tutorial \
      --image-ids imageTag=latest \
      --region us-east-1
```

Step 7: Delete a public repository

If you decide that you no longer need or want an entire repository of images, you can delete the repository. By default, you cannot delete a repository that contains images; however, the `--force` flag allows this. To delete a repository that contains images (and all the images within it), run the following command.

```
aws ecr-public delete-repository \
      --repository-name ecr-tutorial \
      --force \
      --region us-east-1
```

Using the Amazon ECR Public Gallery

The Amazon ECR Public Gallery is a public website to find and share container images hosted in Amazon ECR public repositories. There is no authentication required to browse the public repositories and pull the images. Visit the Amazon ECR Public Gallery at https://gallery.ecr.aws.

Amazon ECR users create public repositories and define the catalog data that appears in the Amazon ECR Public Gallery. The catalog data includes the repository name, a short description, a more detailed description on the **About** tab, and detailed usage instructions on the **Usage** tab. AWS accounts that publish popular or commonly used images can request a **Verified account** badge by contacting support. For more information, see Creating a support case.

The Amazon ECR Public Gallery provides search filters that make it easy to browse through the public repositories. The available search filters include verified accounts, supported operating systems, and system architectures. You can also use the search option to find a specific repository.

You can launch some Amazon ECR Public Gallery container images as web services running on AWS App Runner. When browsing the gallery, look for **Launch to AWS App Runner** on an image's gallery page. An image with this option is a web application that App Runner supports. For more information, see Launch an App Runner service directly from the Amazon ECR Public Gallery in the *App Runner developer guide*.

To get started with creating your own public repository, see Getting started with Amazon ECR Public (p. 4).

Amazon ECR public registries

Amazon ECR public registries host your container images in a highly available and scalable architecture, allowing you to deploy containers reliably for your applications. You can use your public registry to manage public image repositories consisting of Docker and Open Container Initiative (OCI) images. Each AWS account is provided with a default public and private Amazon ECR registry. For information about private registries, see Amazon ECR registries in the *Amazon Elastic Container Registry User Guide.*

Topics

Public registry concepts

The following concepts apply to your public registry.

- A default alias is assigned to your public registry after creating your first public repository. A custom alias can be requested in the public registry settings in the Amazon ECR console.
- Each repository you create in your public registry is available publicly in the Amazon ECR Public Gallery. The Amazon ECR Public Gallery is available at `https://gallery.ecr.aws`. The URL to access a repository in your public registry on the Amazon ECR Public Gallery is `https://gallery.ecr.aws/`*`registry_alias`*`/`*`repository_name`*.

  Note
  Any active alias for your public registry can be used. This includes both the default alias and custom alias for your public registry.
- The URI to use when pulling images from a repository in your public registry is `public.ecr.aws/`*`registry_alias`*`/`*`repository_name`*`:`*`image_tag`*.

  Note
  Any active alias for your public registry can be used. This includes both the default alias and custom alias for your public registry.
- By default, your account has read and write access to the repositories in your private registry. However, users require permissions to make calls to the Amazon ECR APIs and to push images to your repositories. Anyone will be able to pull images from your public repository from the Amazon ECR Public Gallery.

  Important
  If you've previously authenticated to Amazon ECR Public, if your auth token has expired you may receive an authentication error when attempting to do unauthenticated docker pulls from Amazon ECR Public. To resolve this issue, it may be necessary to run `docker logout public.ecr.aws` to avoid the error. This will result in an unauthenticated pull. For more information, see Authentication issues (p. 67).
- You must authenticate your Docker client to your public registry so that you can use the **docker push** command to push images to the repositories in your public registry. For more information, see Registry authentication (p. 13).
- Repositories can be controlled with both IAM user access policies and repository policies. For more information about repository policies, see Public repository policies (p. 27).

Registry authentication

You can use the AWS Management Console, the AWS CLI, or the AWS SDKs to create and manage public repositories. You can also use those methods to perform some actions on images, such as listing or deleting them. These clients use standard AWS authentication methods. Although technically you can use the Amazon ECR Public API to push and pull images, you are much more likely to use the Docker CLI or a language-specific Docker library.

The Docker CLI does not support native IAM authentication methods. Additional steps must be taken so that Amazon ECR can authenticate and authorize Docker push and pull requests.

The following registry authentication methods are available.

Using an authorization token

The permission scope of an authorization token matches that of the IAM principal used to retrieve the authentication token. An authentication token is used to access any Amazon ECR public registry that your IAM principal has access to and is valid for 12 hours. The authentication token is also used to pull any images from a public repository on the Amazon ECR Public Gallery. To obtain an authorization token, you must use the GetAuthorizationToken API operation to retrieve a base64-encoded authorization token containing the username `AWS` and an encoded password. The AWS CLI `get-login-password` command simplifies this by retrieving and decoding the authorization token which you can then pipe into a **docker login** command to authenticate.

To authenticate Docker to an Amazon ECR registry with get-login-password

To authenticate Docker to an Amazon ECR registry with get-login-password, run the **aws ecr-public get-login-password** command. When passing the authentication token to the **docker login** command, use the value `AWS` for the username and specify the Amazon ECR registry URI you want to authenticate to. When authenticating to a public registry, always authenticate to the `us-east-1` Region when using the AWS CLI.

> **Important**
> If you receive an error, install or upgrade to the latest version of the AWS CLI. For more information, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

get-login-password (AWS CLI)

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --
password-stdin public.ecr.aws
```

Using HTTP API authentication

Amazon ECR Public supports the Docker Registry HTTP API, with the exception of the tags API. However, you must provide an authorization token with every HTTP request. You can add an HTTP authorization header using the `-H` option for **curl** and pass the authorization token provided by the **get-authorization-token** AWS CLI command.

To authenticate with the Amazon ECR HTTP API

1. Retrieve an authorization token with the AWS CLI and set it to an environment variable.

```
TOKEN=$(aws ecr-public get-authorization-token --region us-east-1 --output=text --query
 'authorizationData.authorizationToken')
```

2. To authenticate to the API, pass the $TOKEN variable to the -H option of **curl**. For example, the following command lists the manifest details for an image in an Amazon ECR public repository. For more information, see the Docker Registry HTTP API reference documentation.

```
curl -i -H "Authorization: Bearer $TOKEN" https://public.ecr.aws/
v2/registry_alias/repository_name/manifests/image_tag
```

Output:

```
HTTP/1.1 200 OK
Date: Mon, 30 Nov 2020 16:20:09 GMT
Content-Type: application/vnd.docker.distribution.manifest.v2+json
Content-Length: 1569
Connection: keep-alive
Docker-Distribution-Api-Version: registry/2.0

{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 3854,
      "digest": "sha256:2599adbc30c28b1ee5f25a5ebabcc40a37eb81bd89e6f837989ce0fEXAMPLE"
   },
   "layers": [
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 26708056,
         "digest":
 "sha256:f22ccc0b8772d8e1bcb40f137b373686bc27427a70c0e41dd22b3801EXAMPLE"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 850,
         "digest":
 "sha256:3cf8fb62ba5ffb221a2edb2208741346eb4d2d99a174138e4afbb69ceEXAMPLE"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 162,
         "digest":
 "sha256:e80c964ece6a3edf0db1cfc72ae0e6f0699fb776bbfcc92b708fbb945EXAMPLE"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 56322511,
         "digest":
 "sha256:9f379ca76d09bc8d1647896e7dc2d9de21b772fd49cb9f21114de76EXAMPLE"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 190,
         "digest":
 "sha256:d8a5f6eb23fabfe50ebb6facb8c46aa2b2ca0b3a455fe631c312034EXAMPLE"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 207,
```

```
        "digest":
 "sha256:69c1ea2550a94189f95691ed2538c44a2635f988b7cf0d5425f5b4aEXAMPLE"
      }
   ]
}
```

Updating registry settings

Your public registry provides settings to configure a custom alias and display name.

By default, your public registry is assigned a **default alias** after your first public repository is created. You can request a **custom alias** for your registry, and if approved, both the default alias and custom alias can be used to access your public repositories. The registry **display name** appears on each repository in the Amazon ECR Public Gallery as well.

When requesting a custom alias, the following words or phrases should be avoided:

- An alias that includes `aws`, `amazon`, or the name of an AWS service
- An alias using a company name for which you do not have permission to use
- Generic names, such as `test`, `public`, and `marketplace`
- Offensive, inappropriate, or non-inclusive words and phrases

Required IAM permissions

When editing your Amazon ECR public registry settings to request a custom alias, the IAM principal must have permission to call the `ecr-public:PutRegistryAlias` API. This is a private API.

> **Note**
> Setting a **Display name** for your Amazon ECR public registry doesn't require any additional permissions.

The following IAM policy can be added as an inline policy to the principal performing the public registry edit. Replace the example AWS account ID in this example with your own account ID.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ecr-public:PutRegistryAlias",
            "Resource": "arn:aws:ecr-public::123456789012:registry/*"
        }
    ]
}
```

Updating your registry settings

Use the following steps to edit your public registry settings.

To edit public registry settings (AWS Management Console)

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. From the navigation bar, choose the Region to edit your public registry settings in.
3. In the navigation pane, choose **Registries**.

4. On the **Registries** page, select your **Public** registry and then choose **Edit**.
5. For **Custom alias**, enter a custom alias to request.
6. For **Display name**, enter a display name for your registry.
7. Choose **Save changes**.

Amazon ECR public repositories

Amazon Elastic Container Registry provides API operations to create, monitor, and delete public image repositories and set permissions that control who can push images to them. You can perform the same actions in the **Repositories** section of the Amazon ECR console. Amazon ECR integrates with the Docker CLI to push images from your development environments to your public repositories.

A public repository is open to pull images from and is visible on the Amazon ECR Public Gallery. When you create a public repository, you specify catalog data that helps users find and use your images. For more information about the Amazon ECR Public Gallery, see Using the Amazon ECR Public Gallery (p. 11).

Topics

Public repository concepts

- The public repositories that you create with images appear publicly on the Amazon ECR Public Gallery. Visit the Amazon ECR Public Gallery at https://gallery.ecr.aws. For more information, see Using the Amazon ECR Public Gallery (p. 11).
- By default, your account has read and write access to the repositories in your public registry. However, users require permissions to make calls to the Amazon ECR APIs and to push images to your repositories.
- Public repositories can be controlled with both IAM user access policies and repository policies. For more information, see Public repository policies (p. 27).

Creating a public repository

Before you can push your Docker or Open Container Initiative (OCI) images to Amazon ECR, you must create a repository to store them in. Public repositories are visible on the Amazon ECR Public Gallery and are open to publicly pull images from. If you want to create a private repository instead, see Repositories in the *Amazon Elastic Container Registry User Guide*.

To create a public repository (AWS Management Console)

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. From the navigation bar, choose the AWS Region to create your public repository in.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, choose **Create repository**.

5. For **Visibility settings**, choose **Public**.
6. For **Repository name**, enter a unique name for your public repository. The repository name can be specified on its own (for example, `nginx-web-app`) or prepended with a namespace to group the repository into a category (for example, `project-a/nginx-web-app`).
7. For **Repository logo**, choose **Upload file** and select a local image file to use as the repository logo. Amazon ECR uploads your logo as a base64-encoded payload to a publicly available Amazon S3 bucket.

   Note
   The repository logo is only publicly visible in the Amazon ECR Public Gallery for verified accounts.
8. For **Short description** enter a description of the repository. The description field is displayed on the Amazon ECR Public Gallery in the search results and on the repository detail page.
9. For **Content types** select the operating system and system architecture tags to associate with the repository. These tags are publicly displayed in the Amazon ECR Public Gallery as badges on the repository and are used as search filters.
10. For **About**, enter a detailed description for the repository. This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page.
11. For **Usage**, enter details about how to use the images in the repository. This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page.
12. Choose **Create repository**.

Editing a public repository

An existing public repository can be edited to change the catalog data details that are visible in the Amazon ECR Public Gallery.

To edit a repository

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region that the repository to edit is in.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab, and then select the repository to edit and choose **Edit**.
5. For **Repository logo**, if your repository doesn't have a logo, then choose **Upload file** and select a local image file to use as the repository logo. If your repository has a logo currently, choose **Replace file** to choose a new logo file. Choose **Reset** to reset your logo selection.

   Note
   The repository logo is only publicly visible in the Amazon ECR Public Gallery for verified accounts.
6. For **Short description** edit the description of the repository. The description field is displayed on the Amazon ECR Public Gallery in the search results and on the repository detail page.
7. For **Content types** select the operating system and system architecture tags to associate with the repository. These tags are publicly displayed in the Amazon ECR Public Gallery as badges on the repository and are used as search filters.
8. For **About**, enter a detailed description for the repository. This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page. This text must be in the GitHub Flavored Markdown format. For examples, see Repository catalog data (p. 19).
9. For **Usage**, enter details about how to use the images in the repository. This field is publicly visible on the Amazon ECR Public Gallery on the repository detail page. This text must be in the GitHub Flavored Markdown format. For examples, see Repository catalog data (p. 19).

10. Choose **Save** to update the repository settings.

Repository catalog data

When you create a public repository, you specify the catalog data that helps users find, understand, and use the images in the repository. The catalog data that you configure for a public repository includes a short description, the operating system and system architecture compatibilities, an optional logo, an **About** section that provides a more detailed description, and an **Usage** section that provides details on how to use the images.

When you specify a logo, the logo is specified as a blob that's a base64-encoded string. The supported image dimensions for both height and width must be at least 60 pixels but can be up to 2048 pixels long. The maximum file size is 500 KB. To generate a blob from an existing PNG file, run the following command.

```
cat myrepoimage.png | base64
```

The text for the **About** and **Usage** must be in the GitHub Flavored Markdown format. When using the API, SDK, or AWS CLI to format the text, use `/n` to indicate a line break.

The following table provides examples for specifying certain element types in the `About` and `Usage` sections of your repository catalog data.

Examples

The following are examples of how to format the **About** and **Usage** repository catalog data so that it appears properly on the Amazon ECR Public Gallery.

Topics

Example: Headings

Headings are designated by the number sign (#). A single number sign and a space indicate a top-level heading, two number signs create a second-level heading, and three number signs create a third-level heading. This is illustrated in the following examples.

AWS Management Console

The following example is the format for headings in the console.

```
# Heading level one

Body text
```

```
## Heading level two

Body text

### Heading level three

Body text
```

AWS CLI

The following example is the format to use for headings in the AWS CLI.

```
# Heading level one\n\nBody text\n\n## Heading level two\n\nBody text\n\n### Heading level
 three\n\nBody text\n\n#### Heading level four\n\nBody text
```

Example: Text formatting

You can use the following syntax to apply italics, bold, or strikethrough to text. The syntax is the same for both the console and the AWS CLI.

```
*This text appears in italics*
```

```
**This text appears in bold**
```

```
~~This text appears in strikethrough~~
```

Example: Code formatting

You can use the following syntax for single-line and multi-line code blocks. The syntax is the same for both the console and the AWS CLI.

```
`code text`
```

```
```
multi-line
codeblock
```
```

Example: Links

You can create a clickable web link by surrounding the `link_text` with square brackets and surrounding the full URL with parentheses. The syntax for is the same for both the console and the AWS CLI.

```
[What is Amazon Elastic Container Registry?](https://docs.aws.amazon.com/AmazonECR/latest/
userguide/what-is-ecr.html)
```

Example: Lists

To format lines as part of a bulleted list, enter them on separate lines with a single asterisk and then a space at the beginning of the line. To format lines as part of a numbered list, enter them on separate lines with a number, period, and space at the beginning of the line.

AWS Management Console

The following example is the format to use for lists in the console.

```
* Bullet 1
* Bullet 2
* Bullet 3
```

```
1. Step one
2. Step two
3. Step three
```

AWS CLI

The following example is the format to use for lists in the AWS CLI.

```
* Bullet 1\n* Bullet 2\n* Bullet 3
```

```
1. Step one\n2. Step two\n3. Step three
```

Example: Full **About** description

The following screenshot from the Amazon ECR Public Gallery displays how an **About** section is constructed. This section covers the format to use for this text when using both the AWS Management Console and the AWS CLI.

About | Usage | Image tags

Quick reference

Maintained by: the Amazon Linux Team

Where to get help: the Docker Community Forums, the Docker Community S

Supported tags and respective `dockerfile` links

- `2.0.20200722.0`, `2`, `latest`
- `2.0.20200722.0-with-sources`, `2-with-sources`, `with-sources`
- `2018.03.0.20200602.1`, `2018.03`, `1`
- `2018.03.0.20200602.1-with-sources`, `2018.03-with-sources`,

What is Amazon Linux?

Amazon Linux is provided by Amazon Web Services. It is designed to provide running on Amazon EC2. The full distribution includes packages that enable tools and many popular Amazon Web Services libraries and tools. Amazon W instances running Amazon Linux.

The Amazon Linux container image contains a minimal set of packages. To in

Amazon Web Services provides two versions of Amazon Linux: Amazon Linux

For information on security updates for Amazon Linux, please refer to Amaz that Docker Hub's vulnerability scanning for Amazon Linux is currently based vulnerabilities.

Where can I run Amazon Linux container images?

You can run Amazon Linux container images in any Docker based environme clusters.

License

Amazon Linux is available under the GNU General Public License, version 2.0

`-qi [package name]` or check `/usr/share/doc/[package name]-*` a

AWS Management Console

The following is the format to use for the preceding screenshot in the console.

```
## Quick reference

Maintained by: [the Amazon Linux Team](https://github.com/aws/amazon-linux-docker-images)

Where to get help: [the Docker Community Forums](https://forums.docker.com/), [the Docker
 Community Slack](https://dockr.ly/slack), or [Stack Overflow](https://stackoverflow.com/
search?tab=newest&q=docker)

## Supported tags and respective `dockerfile` links

* [`2.0.20200722.0`, `2`, `latest`](https://github.com/amazonlinux/container-images/
blob/03d54f8c4d522bf712cffd6c8f9aafba0a875e78/Dockerfile)
* [`2.0.20200722.0-with-sources`, `2-with-sources`, `with-sources`](https://github.com/
amazonlinux/container-images/blob/1e7349845e029a2e6afe6dc473ef17d052e3546f/Dockerfile)
* [`2018.03.0.20200602.1`, `2018.03`, `1`](https://github.com/amazonlinux/container-images/
blob/f10932e08c75457eeb372bf1cc47ea2a4b8e98c8/Dockerfile)
* [`2018.03.0.20200602.1-with-sources`, `2018.03-with-sources`, `1-with-sources`](https://
github.com/amazonlinux/container-images/blob/8c9ee491689d901aa72719be0ec12087a5fa8faf/
Dockerfile)

## What is Amazon Linux?

Amazon Linux is provided by Amazon Web Services. It is designed to provide a stable,
 secure, and high-performance execution environment for applications running on Amazon
 EC2. The full distribution includes packages that enable easy integration with Amazon Web
 Services, including launch configuration tools and many popular AWS libraries and tools.
 AWS provides ongoing security and maintenance updates to all instances running Amazon
 Linux.

The Amazon Linux container image contains a minimal set of packages. To install additional
 packages, [use `yum`](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-
software.html).

Amazon Web Services provides two versions of Amazon Linux: [Amazon Linux 2](https://
aws.amazon.com/amazon-linux-2/) and [Amazon Linux AMI](https://aws.amazon.com/amazon-linux-
ami/).

For information on security updates for Amazon Linux, please refer to [Amazon Linux
 2 Security Advisories](https://alas.aws.amazon.com/alas2.html) and [Amazon Linux AMI
 Security Advisories](https://alas.aws.amazon.com/). Note that Docker Hub's vulnerability
 scanning for Amazon Linux is currently based on RPM versions, which does not reflect the
 state of backported patches for vulnerabilities.

## Where can I run Amazon Linux container images?

You can run Amazon Linux container images in any Docker based environment. Examples
 include, your laptop, in Amazon EC2 instances, and Amazon ECS clusters.

## License

Amazon Linux is available under the [GNU General Public License, version 2.0](https://
github.com/aws/amazon-linux-docker-images/blob/master/LICENSE). Individual software
 packages are available under their own licenses; run `rpm -qi [package name]` or check `/
usr/share/doc/[package name]-*` and `/usr/share/licenses/[package name]-*` for details.

As with all Docker images, these likely also contain other software which may be under
 other licenses (such as Bash, etc from the base distribution, along with any direct or
 indirect dependencies of the primary software being contained).
```

```
Some additional license information which was able to be auto-detected might be found in
 [the `repo-info` repository's `amazonlinux/` directory](https://github.com/docker-library/
repo-info/tree/master/repos/amazonlinux).

## Security

For information on security updates for Amazon Linux, please refer to [Amazon Linux
 2 Security Advisories](https://alas.aws.amazon.com/alas2.html) and [Amazon Linux AMI
 Security Advisories](https://alas.aws.amazon.com/). Note that Docker Hub's vulnerability
 scanning for Amazon Linux is currently based on RPM versions, which does not reflect the
 state of backported patches for vulnerabilities.
```

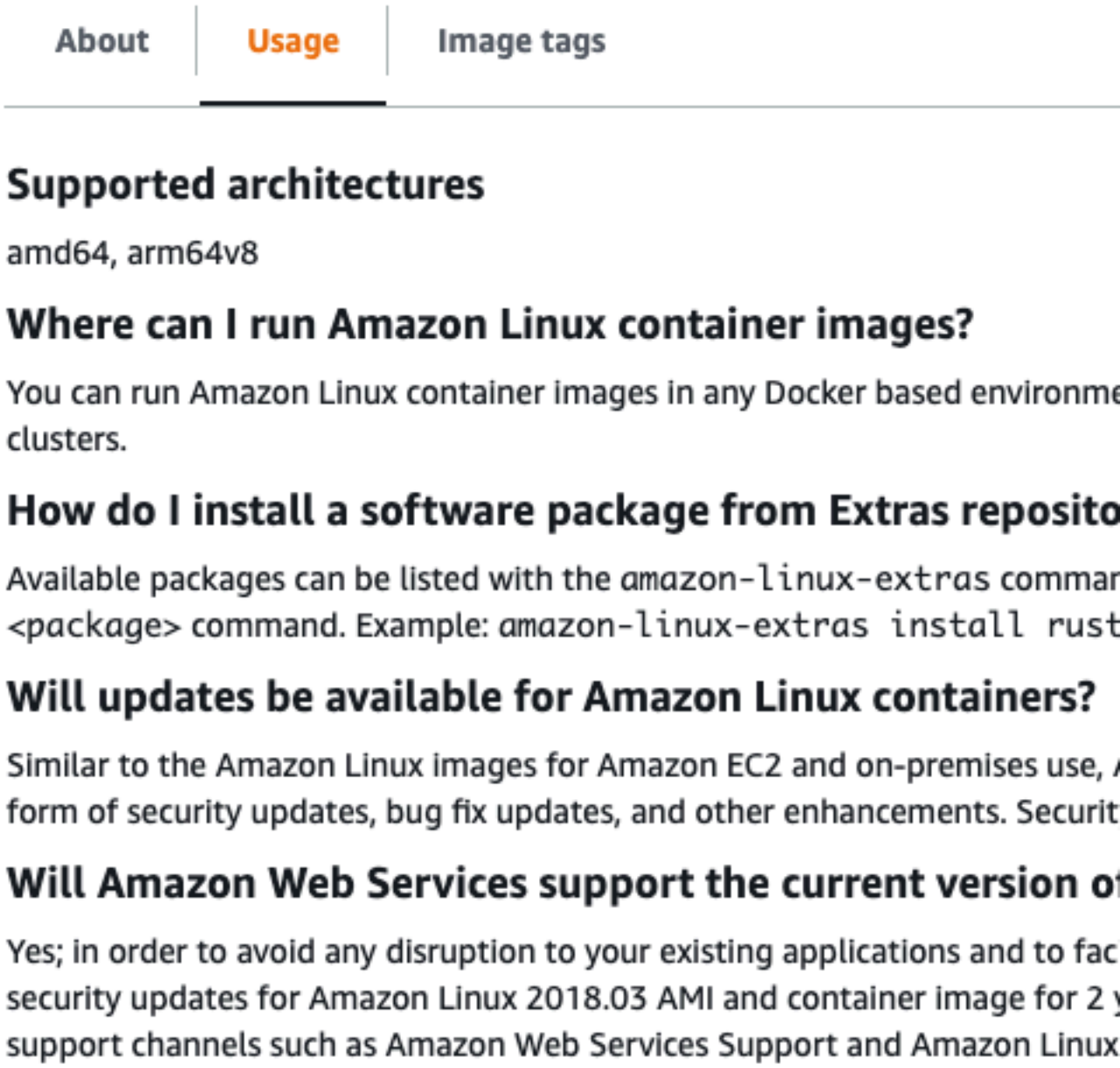
AWS CLI

The following is the format to use for the preceding screenshot in the AWS CLI.

```
## Quick reference\n\nMaintained by: [the Amazon Linux Team](https://github.com/aws/
amazon-linux-docker-images)\n\nWhere to get help: [the Docker Community Forums](https://
forums.docker.com/), [the Docker Community Slack](https://dockr.ly/slack), or [Stack
 Overflow](https://stackoverflow.com/search?tab=newest&q=docker)\n\n## Supported tags and
 respective `dockerfile` links\n\n* [`2.0.20200722.0`, `2`, `latest`](https://github.com/
amazonlinux/container-images/blob/03d54f8c4d522bf712cffd6c8f9aafba0a875e78/Dockerfile)\n*
 [`2.0.20200722.0-with-sources`, `2-with-sources`, `with-sources`](https://github.com/
amazonlinux/container-images/blob/1e7349845e029a2e6afe6dc473ef17d052e3546f/Dockerfile)\n*
 [`2018.03.0.20200602.1`, `2018.03`, `1`](https://github.com/amazonlinux/container-images/
blob/f10932e08c75457eeb372bf1cc47ea2a4b8e98c8/Dockerfile)\n* [`2018.03.0.20200602.1-
with-sources`, `2018.03-with-sources`, `1-with-sources`](https://github.com/amazonlinux/
container-images/blob/8c9ee491689d901aa72719be0ec12087a5fa8faf/Dockerfile)\n\n## What is
 Amazon Linux?\n\nAmazon Linux is provided by Amazon Web Services (AWS). It is designed
 to provide a stable, secure, and high-performance execution environment for applications
 running on Amazon EC2. The full distribution includes packages that enable easy
 integration with AWS, including launch configuration tools and many popular AWS libraries
 and tools. AWS provides ongoing security and maintenance updates to all instances
 running Amazon Linux.\n\nThe Amazon Linux container image contains a minimal set of
 packages. To install additional packages, [use `yum`](https://docs.aws.amazon.com/AWSEC2/
latest/UserGuide/managing-software.html).\n\nAWS provides two versions of Amazon Linux:
 [Amazon Linux 2](https://aws.amazon.com/amazon-linux-2/) and [Amazon Linux AMI](https://
aws.amazon.com/amazon-linux-ami/).\n\nFor information on security updates for Amazon
 Linux, please refer to [Amazon Linux 2 Security Advisories](https://alas.aws.amazon.com/
alas2.html) and [Amazon Linux AMI Security Advisories](https://alas.aws.amazon.com/).
 Note that Docker Hub's vulnerability scanning for Amazon Linux is currently based on RPM
 versions, which does not reflect the state of backported patches for vulnerabilities.\n
\n## Where can I run Amazon Linux container images?\n\nYou can run Amazon Linux container
 images in any Docker based environment. Examples include, your laptop, in Amazon EC2
 instances, and Amazon ECS clusters.\n\n## License\n\nAmazon Linux is available under the
 [GNU General Public License, version 2.0](https://github.com/aws/amazon-linux-docker-
images/blob/master/LICENSE). Individual software packages are available under their own
 licenses; run `rpm -qi [package name]` or check `/usr/share/doc/[package name]-*` and
 `/usr/share/licenses/[package name]-*` for details.\n\nAs with all Docker images, these
 likely also contain other software which may be under other licenses (such as Bash, etc
 from the base distribution, along with any direct or indirect dependencies of the primary
 software being contained).\n\nSome additional license information which was able to be
 auto-detected might be found in [the `repo-info` repository's `amazonlinux/` directory]
(https://github.com/docker-library/repo-info/tree/master/repos/amazonlinux).\n\n## Security
\n\nFor information on security updates for Amazon Linux, please refer to [Amazon Linux
 2 Security Advisories](https://alas.aws.amazon.com/alas2.html) and [Amazon Linux AMI
 Security Advisories](https://alas.aws.amazon.com/). Note that Docker Hub's vulnerability
 scanning for Amazon Linux is currently based on RPM versions, which does not reflect the
 state of backported patches for vulnerabilities.
```

Example: Full **Usage** description

The following screenshot from the Amazon ECR Public Gallery displays how an **Usage** section is constructed. This section covers how to format this text using both the AWS Management Console and the AWS CLI.



AWS Management Console

The following is the format to use for the preceding screenshot in the console.

```
## Supported architectures

amd64, arm64v8
```

```
## Where can I run Amazon Linux container images?

You can run Amazon Linux container images in any Docker based environment. Examples
 include, your laptop, in Amazon EC2 instances, and Amazon ECS clusters.

## How do I install a software package from Extras repository in Amazon Linux 2?

Available packages can be listed with the `amazon-linux-extras` command. Packages can be
 installed with the `amazon-linux-extras install <package>` command. Example: `amazon-
linux-extras install rust1`

## Will updates be available for Amazon Linux containers?

Similar to the Amazon Linux images for Amazon EC2 and on-premises use, Amazon Linux
 container images will get ongoing updates from Amazon in the form of security updates, bug
 fix updates, and other enhancements. Security bulletins for Amazon Linux are available at
 https://alas.aws.amazon.com/

## Will Amazon Web Services support the current version of Amazon Linux going forward?

Yes; in order to avoid any disruption to your existing applications and to facilitate
 migration to Amazon Linux 2, AWS will provide regular security updates for Amazon Linux
 2018.03 AMI and container image for 2 years after the final LTS build is announced. You
 can also use all your existing support channels such as AWS Support and Amazon Linux
 Discussion Forum to continue to submit support requests.
```

AWS CLI

The following is the format to use for the preceding screenshot in the AWS CLI.

```
## Supported architectures\n\namd64, arm64v8\n\n## Where can I run Amazon Linux container
 images?\n\nYou can run Amazon Linux container images in any Docker based environment.
 Examples include, your laptop, in Amazon EC2 instances, and ECS clusters.\n\n## How do I
 install a software package from Extras repository in Amazon Linux 2?\n\nAvailable packages
 can be listed with the `amazon-linux-extras` command. Packages can be installed with the
 `amazon-linux-extras install <package>` command. Example: `amazon-linux-extras install
 rust1`\n\n## Will updates be available for Amazon Linux containers?\n\nSimilar to the
 Amazon Linux images for Amazon EC2 and on-premises use, Amazon Linux container images
 will get ongoing updates from Amazon in the form of security updates, bug fix updates,
 and other enhancements. Security bulletins for Amazon Linux are available at https://
alas.aws.amazon.com/\n\n## Will AWS Support the current version of Amazon Linux going
 forward?\n\nYes; in order to avoid any disruption to your existing applications and to
 facilitate migration to Amazon Linux 2, AWS will provide regular security updates for
 Amazon Linux 2018.03 AMI and container image for 2 years after the final LTS build is
 announced. You can also use all your existing support channels such as AWS Support and
 Amazon Linux Discussion Forum to continue to submit support requests.
```

Viewing public repository information

After you create a public repository, you can view details about it in the AWS Management Console. For each image in the repository, you can view the image size, the URI for pulling the image, the SHA digest, the image tags, and the time when the image was last pushed. You can review the catalog data for the Amazon ECR Public Gallery. You can also see the repository permission policies that are associated with the repository.

> **Note**
> Beginning with Docker version 1.9, the Docker client compresses image layers before pushing them to a V2 Docker registry. The output of the **docker images** command shows the uncompressed image size. Therefore, the image size that's returned might be larger than the image sizes that are shown in the AWS Management Console.

To view public repository information

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region that the repository to view is in.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab, and then choose the repository to view the details of.
5. On the **Repositories > *repository_name*** page, choose **View public listing** to navigate to the repository detail page in the Amazon ECR Public Gallery in a new tab or use the navigation bar to view more details about the repository.
   - Choose **Images** to view information about the images in the repository. If there are untagged images that you want to delete, you can select the box to the left of the repositories to delete and choose **Delete**. For more information, see Deleting an image in a public repository (p. 39).
   - Choose **Gallery detail** to view the public catalog data for the repository.
   - Choose **Permissions** to view the repository policies that are applied to the repository. For more information, see Public repository policies (p. 27).

Deleting a public repository

If you're finished using a repository, you can delete it. When you delete a repository in the AWS Management Console, all of the images that are contained in the repository are also deleted. This action can't be undone.

To delete a public repository

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the AWS Region that contains the repository to delete.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab, and then select the repository to delete and choose **Delete**.
5. In the **Delete *repository_name*** window, double check the repositories that you selected to delete and choose **Delete**.

   Important
   Any images in the selected repositories are also deleted.

Public repository policies

Amazon ECR uses resource-based permissions to control access to public repositories. With resource-based permissions, you specify which users, or roles have access to a public repository and what actions they can perform on it. By default, only the repository owner has access to a repository. With public repositories, anyone can pull images from the repository. However, only the repository owner has access to push to the repository. You can apply a policy document to allow additional permissions to your repository.

Repository policies vs IAM policies

Amazon ECR public repository policies are a subset of IAM policies that are both scoped for and specifically used for controlling access to individual Amazon ECR repositories. In general, you use IAM

policies to apply permissions for the entire Amazon ECR service. However, you can also use IAM policies to control access to specific resources.

For determining which actions a specific IAM user or role might perform on a repository, you use both Amazon ECR repository policies and IAM policies. If a user or role is allowed to perform an action through a repository policy but is denied permission through an IAM policy, the action is denied. Similarly, if a user or role is denied permission through an IAM policy even though that identity is allowed to perform an action, the action is denied. You can grant a user or role permission for an action through either a repository policy or an IAM policy, but you can't grant permission both ways.

> **Important**
> Amazon ECR requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to a registry and push any images to an Amazon ECR repository.

You can use either of these policy types to control access to your public repositories, as shown in the following examples.

This example shows an Amazon ECR public repository policy, which allows for a specific IAM user to describe the repository and the images within the repository.

```
{
  "Version": "2008-10-17",
  "Statement": [{
    "Sid": "ECR Public Repository Policy",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::account-id:user/username"
    },
    "Action": [
       "ecr-public:DescribeImages",
       "ecr-public:DescribeRepositories"
    ]
  }]
}
```

This example shows an IAM policy that achieves the same goal as the preceding example. In this example, the policy is scoped to a public repository (specified by the full Amazon Resource Name (ARN) of the public repository) using the resource parameter. For more information about ARN format, see Resources (p. 47).

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ECR Public Repository Policy",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::account-id:user/username"
    },
    "Action": [
      "ecr-public:DescribeImages",
      "ecr-public:DescribeRepositories"
    ],
    "Resource": [
      "arn:aws:ecr-public::account-id:repository/repository-name"
    ]
    }]
}
```

Topics

Setting a repository policy statement

You can add an access policy statement to a public repository in the AWS Management Console by following these steps. You can add multiple policy statements per public repository. For example policies, see Public repository policy examples (p. 30).

> **Important**
> Amazon ECR requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to a registry and push any images to an Amazon ECR repository.

To set a repository policy statement

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the AWS Region that contains the repository to set a policy statement on.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab, and then choose the repository to set a policy statement on.
5. In the navigation pane, choose **Permissions**, **Edit**.
6. On the **Edit permissions** page, choose **Add statement**.
7. For **Statement name**, enter a name for the statement.
8. For **Effect**, choose whether the policy statement results in an allow or an explicit deny.
9. For **Principal**, choose the scope to apply the policy statement to. For more information, see AWS JSON Policy Elements: Principal in the *IAM User Guide*.
   - You can apply the statement to all authenticated AWS users by selecting the **Everyone (*)** check box.
   - For **Service principal**, specify the service principal name (for example, `ecs.amazonaws.com`) to apply the statement to a specific service.
   - For **AWS Account IDs**, specify an AWS account number (for example, `111122223333`) to apply the statement to all users under a specific AWS account. Multiple accounts can be specified by using a comma-separated list.
   - For **IAM Entities**, select the roles or users under your AWS account to apply the statement to.

     > **Note**
     > For more complicated repository policies that are not currently supported in the AWS Management Console, you can apply the policy with the **set-repository-policy** AWS CLI command.
10. For **Actions**, choose the scope of the Amazon ECR API operations that the policy statement applies to from the list of individual API operations.
11. When you're finished, choose **Save** to set the policy.
12. Repeat the previous step for each repository policy to add.

Deleting a public repository policy statement

If you no longer want an existing repository policy statement to apply to a repository, you can delete it.

To delete a repository policy statement

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region that contains the repository to delete a policy statement from.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab, and then choose the repository to delete a policy statement from.
5. In the navigation pane, choose **Permissions**, **Edit**.
6. On the **Edit permissions** page, choose **Delete**.

Public repository policy examples

The following examples show policy statements that you use to control the permissions that users have to your public repositories.

> **Important**
> Amazon ECR requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to a registry and push any images to an Amazon ECR repository.

Example: Allow an IAM user within your account

The following repository policy allows usres within your account to push images.

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

Example: Allow another account

The following repository policy allows a specific account to push images.

```
{
    "Version": "2008-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowCrossAccountPush",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account-id:root"
            },
            "Action": [
                "ecr-public:BatchCheckLayerAvailability",
                "ecr-public:PutImage",
                "ecr-public:InitiateLayerUpload",
                "ecr-public:UploadLayerPart",
                "ecr-public:CompleteLayerUpload"
            ]
        }
    ]
}
```

Tagging an Amazon ECR Public repository

To help you manage your Amazon ECR Public repositories, you can optionally assign your own metadata to each repository by using *tags*. This topic provides an overview about tags and how to create them.

Contents

Tag basics

A tag is a label that you assign to an AWS resource. Each tag consists of a *key* and an optional *value*. You define both of them.

You can use tags to categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type. This is because you can quickly identify a specific resource based on the tags you've assigned to it. For example, you can define a set of tags for your account's Amazon ECR Public repositories to track each repository's owner.

We recommend that you devise a set of tag keys that meets your specific needs. Using a consistent set of tag keys can help you keep better track of your resources and find specific resources quickly. That is, you can search and filter the resources based on the specific tags that you add.

Tags don't have any semantic meaning to Amazon ECR and are interpreted strictly as a string of characters. Tags aren't automatically assigned to your resources. You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string. However, you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags for the resource are also deleted.

You can work with tags using the AWS Management Console, the AWS CLI, and the Amazon ECR Public API.

If you're using AWS Identity and Access Management (IAM), you can control which users in your AWS account have permission to manage tags.

Tagging your resources

You can tag new or existing Amazon ECR Public repositories.

If you're using the Amazon ECR console, you can apply tags to new resources when they're created or to existing resources by using the **Tags** option on the navigation pane at any time.

If you're using the Amazon ECR Public API, the AWS CLI, or an AWS SDK, you can apply tags to new repositories using the `tags` parameter on the `CreateRepository` API action or use the `TagResource` API action to apply tags to existing resources. For more information, see TagResource.

Additionally, if tags can't be applied when a repository is created, the repository creation process is rolled back. This ensures that repositories are either created with tags or not created at all and that no repositories are left untagged at any time. By tagging repositories when they're created, you eliminate the need to run custom tagging scripts after the repository is created.

Tag restrictions

The following restrictions and conditions apply to tags:

- A maximum of 50 tags can be associated with a repository.
- For each repository, each tag key must be unique, and each tag key can have only one value.
- A key can be up to 128 Unicode characters long, and must be in UTF-8.
- A value can be up to 256 Unicode characters long, and must be in UTF-8.
- If your tagging schema is used across multiple services and resources, remember that other services might have restrictions on allowed characters. Characters that are allowed across most services are letters, numbers, white spaces (representable in UTF-8), plus signs (+), hyphens (-), equal signs (=), periods (.) underscores (_), colons (:) slashes (/), and the at symbol (@).
- Tag keys and values are case sensitive.
- Don't use the `aws:` prefix for either keys or values. It's reserved for AWS use only. You can't edit or delete tag keys or values with this prefix. Tags with this prefix aren't included in your tag per resource quota.

Working with tags using the console

Using the Amazon ECR console, you can manage the tags that are associated with new or existing repositories.

When you select a specific repository in the Amazon ECR console, you can view the tags by selecting **Tags** in the navigation pane.

To add a tag to a public repository

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. From the navigation bar, select the AWS Region to use.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, on the **Public** tab, choose the repository to view.
5. On the **Repositories >** ***repository_name*** page, select **Tags** from the navigation pane.
6. On the **Tags** page, select **Add tags**, **Add tag**.
7. On the **Edit Tags** page, specify the key and value for each tag, and then choose **Save**.

To delete a tag from an individual resource

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
2. From the navigation bar, select the Region to use.
3. On the **Repositories** page, on the **Public** tab, choose the repository to view.
4. On the **Repositories > *repository_name*** page, select **Tags** from the navigation pane.
5. On the **Tags** page, select **Edit**.
6. On the **Edit tags** page, select **Remove** for each tag you want to delete, and choose **Save**.

Working with tags using the AWS CLI or API

Use the following to add, update, list, and delete the tags for your resources. The corresponding documentation provides examples.

Tagging support for Amazon ECR Public resources

Task	AWS CLI	API action
Add or overwrite one or more tags.	tag-resource	TagResource
Delete one or more tags.	untag-resource	UntagResource

The following examples show how to manage tags using the AWS CLI.

Example 1: Tag an existing public repository

The following command tags an existing public repository.

```
aws ecr-public tag-resource \
      --resource-arn arn:aws:ecr-public::account_id:repository/repository_name \
      --tags Key=stack,Value=dev \
      --region us-east-1
```

Example 2: Tag an existing public repository with multiple tags

The following command tags an existing repository.

```
aws ecr-public tag-resource \
      --resource-arn arn:aws:ecr-public::account_id:repository/repository_name \
      --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3 \
      --region us-east-1
```

Example 3: Untag an existing public repository

The following command deletes a tag from an existing public repository.

```
aws ecr-public untag-resource \
      --resource-arn arn:aws:ecr-public::account_id:repository/repository_name \
      --tag-keys tag_key \
      --region us-east-1
```

Example 4: List tags for a public repository

The following command lists the tags that are associated with an existing public repository.

```
aws ecr-public list-tags-for-resource \
      --resource-arn arn:aws:ecr-public::account_id:repository/repository_name \
      --region us-east-1
```

Example 5: Create a public repository and apply a tag

The following command creates a public repository that's named `test-repo` and adds a tag with key `team` and value `devs`.

```
aws ecr-public create-repository \
      --repository-name test-repo \
      --tags Key=team,Value=devs \
      --region us-east-1
```

Amazon ECR Public images

Amazon ECR Public stores Docker images, Open Container Initiative (OCI) images, and OCI compatible artifacts in repositories. You can use the Docker CLI or your preferred client to push and pull images to and from your repositories.

> **Important**
> Amazon ECR requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to a registry and push any images to an Amazon ECR repository.

Topics

Pushing an image to a public repository

You can push your Docker images to an Amazon ECR public repository with the **docker push** command.

> **Important**
> Amazon ECR requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to a registry and push any images to an Amazon ECR repository.

Amazon ECR Public also supports creating and pushing Docker manifest lists which are used for multi-architecture images. Each image referenced in a manifest list must already be pushed to your repository. For more information, see Pushing a multi-architecture image to a public repository (p. 36).

To push a Docker image to an Amazon ECR public repository

1. Authenticate your Docker client to the Amazon ECR public registry to which you intend to push your image. Authentication tokens are valid for 12 hours. For more information, see Registry authentication (p. 13).
2. If your public repository does not exist in the registry you intend to push to yet, create it. For more information, see Creating a public repository (p. 17).
3. Identify the image to push. Run the **docker images** command to list the images on your system.

```
docker images
```

   You can identify an image with the *repository:tag* value or the image ID in the resulting command output.

4. Tag your image with the Amazon ECR public registry, public repository, and optional image tag name combination to use. The public registry format is `public.ecr.aws/`*`registry_alias`*. The public repository name should match the repository that you created for your image. If you omit the image tag, we assume that the tag is `latest`.

   The following example tags an image with the ID *`e9ae3c220b23`* as `public.ecr.aws/`*`registry_alias`*`/my-web-app`

   ```
   docker tag e9ae3c220b23 public.ecr.aws/registry_alias/my-web-app
   ```

5. Push the image using the **docker push** command:

   ```
   docker push public.ecr.aws/registry_alias/my-web-app
   ```

6. (Optional) Apply any additional tags to your image and push those tags to Amazon ECR Public by repeating Step 4 (p. 36) and Step 5 (p. 36). You can apply up to 1000 tags per image in Amazon ECR Public.

Pushing a multi-architecture image to a public repository

Amazon ECR Public supports creating and pushing Docker manifest lists which are used for multi-architecture images. A *manifest list* is a list of images that is created by specifying one or more image names. Typically the manifest list is created from images that serve the same function but for different operating systems or architectures, but this is not required. For more information, see docker manifest.

> **Important**
> Your Docker CLI must have experimental features enabled to use this feature. For more information, see Experimental features.

A manifest list can be pulled or referenced in an Amazon ECS task definition or Amazon EKS pod spec like other Amazon ECR Public images.

The following steps can be used to create and push a Docker manifest list to an Amazon ECR public repository. You must already have the images pushed to your public repository to reference in the Docker manifest. For information on pushing an image, see Pushing an image to a public repository (p. 35).

To push a multi-architecture Docker image to an Amazon ECR public repository

1. Authenticate your Docker client to the Amazon ECR public registry to which you intend to push your image. Authentication tokens are valid for 12 hours. For more information, see Registry authentication (p. 13).
2. List the images in your public repository, confirming the image tags.

   ```
   aws ecr-public describe-images \
        --repository-name my-web-app
        --region us-east-1
   ```

3. Create the Docker manifest list. The `manifest create` command verifies that the referenced images are already in your public repository and creates the manifest locally.

   ```
   docker manifest create public.ecr.aws/registry_alias/my-web-
   app public.ecr.aws/registry_alias/my-web-app:image_one_tag
    public.ecr.aws/registry_alias/my-web-app:image_two
   ```

4. (Optional) Inspect the Docker manifest list. This enables you to confirm the size and digest for each image manifest referenced in the manifest list.

```
docker manifest inspect public.ecr.aws/registry_alias/my-web-app
```

5. Push the Docker manifest list to your Amazon ECR public repository.

```
docker manifest push public.ecr.aws/registry_alias/my-web-app
```

Pushing a Helm chart to a public repository

Amazon ECR Public supports pushing Open Container Initiative (OCI) artifacts to your public repositories. To display this functionality, use the following steps to push a Helm chart to Amazon ECR Public.

To push a Helm chart to an Amazon ECR public repository

1. Install Helm version `3.8.0` or higher of the Helm client. These steps were written using Helm version `3.8.0`. For more information, see Installing Helm.
2. Use the following steps to create a test Helm chart. For more information, see Helm Docs - Getting Started.

   a. Create a Helm chart named `helm-test-chart` and clear the contents of the `templates` directory.

   ```
   helm create helm-test-chart
   rm -rf ./helm-test-chart/templates/*
   ```

   b. Create a ConfigMap in the `templates` folder.

   ```
   cd helm-test-chart/templates
   cat <<EOF > configmap.yaml
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: helm-test-chart-configmap
   data:
     myvalue: "Hello World"
   EOF
   ```

3. Package the chart. The output will contain the filename of the packaged chart which you use when pushing the Helm chart.

   ```
   cd ..
   helm package helm-test-chart
   ```

   Output

   ```
   Successfully packaged chart and saved it to: /Users/username/helm-test-chart-0.1.0.tgz
   ```

4. Create a public repository to store your Helm chart. For more information, see Creating a public repository (p. 17).

   ```
   aws ecr-public create-repository \
        --repository-name helm-test-chart \
        --region us-east-1
   ```

5. Authenticate your Helm client to the Amazon ECR public registry to which you intend to push your Helm chart. Authentication tokens are valid for 12 hours. For more information, see Registry authentication (p. 13).

```
aws ecr-public get-login-password \
     --region us-east-1 | helm registry login \
     --username AWS \
     --password-stdin public.ecr.aws
```

6. Push the Helm chart using the **helm push** command. The output should include the Amazon ECR repository URI and SHA digest.

```
helm push helm-test-chart-0.1.0.tgz oci://public.ecr.aws/registry_alias
```

7. Describe your Helm chart.

```
aws ecr-public describe-images \
     --repository-name helm-test-chart \
     --region us-east-1
```

```
{
    "imageDetails": [
        {
            "registryId": "aws_account_id",
            "repositoryName": "helm-test-chart",
            "imageDigest":
 "sha256:f23ab9dc0fda33175e465bd694a5f4cade93eaf62715fa9390d9fEXAMPLE",
            "imageTags": [
                "0.1.0"
            ],
            "imageSizeInBytes": 1636,
            "imagePushedAt": "2022-06-01T12:17:39-05:00",
            "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json"
        }
    ]
}
```

Pulling an image from the Amazon ECR Public Gallery

If you would like to run a Docker image that is available in Amazon ECR Public, you can pull it to your local environment with the **docker pull** command. You can do this from any public repository. Every public repository created on Amazon ECR Public is available on the Amazon ECR Public Gallery. Visit the Amazon ECR Public Gallery at https://gallery.ecr.aws. For more information, see Using the Amazon ECR Public Gallery (p. 11).

Amazon ECR Public supports both unauthenticated and authenticated pulls from public repositories. There are separate service quotas for each type of image pull. For more information, see Amazon ECR Public service quotas (p. 64).

- An unauthenticated pull is a pull without an auth token. You can confirm whether there is an auth token in your Docker configuration by checking your `~/.docker/config.json` file. If you've previously authenticated to Amazon ECR Public but you want to perform an unauthenticated pull, you can logout using the `docker logout public.ecr.aws` command which will remove the auth token from your Docker configuration file.

- An authenticated pull requires that you authenticate to Amazon ECR Public prior to the pull request. For more information, see Registry authentication (p. 13).

  Note
  For authenticated pulls, Amazon ECR Public requires that users have permission to make calls to the `ecr-public:GetAuthorizationToken` and `sts:GetServiceBearerToken` API through an IAM policy before they can authenticate to Amazon ECR Public and pull an image from a public repository.

To pull a public image from the Amazon ECR Public Gallery

1. Identify the image to pull. You can view the available public repositories on the Amazon ECR Public Gallery at https://gallery.ecr.aws.
2. For authenticated pulls, you must authenticate your Docker client to the Amazon ECR public registry. Authentication tokens are valid for 12 hours. For more information, see Registry authentication (p. 13).

   Note
   For unauthenticated pulls, you can skip this step.
3. Pull the image using the **docker pull** command. The image name format should be *registry_alias*/*repository*[:*tag*] to pull by tag, or *registry_alias*/*repository*[@*digest*] to pull by digest.

```
docker pull public.ecr.aws/registry_alias/repository:tag
```

Deleting an image in a public repository

If you are done using an image, you can delete it from your public repository. You can delete an image using the AWS Management Console, or the AWS CLI.

Note
If you are done with a public repository, you can delete the entire repository and all of the images within it. For more information, see Deleting a public repository (p. 27).

To delete a public image with the AWS Management Console

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region that contains the image to delete.
3. In the navigation pane, choose **Repositories**.
4. On the **Repositories** page, select the **Public** tab and then select the repository containing the image to delete.
5. On the **Repositories: *repository_name*** page, select the box to the left of the image to delete and choose **Delete**.
6. In the **Delete image(s)** dialog box, verify that the selected images should be deleted and choose **Delete**.

To delete a public image with the AWS CLI

1. List the image tags in your repository.

```
aws ecr-public describe-image-tags \
      --repository-name my-repo \
```

```
      --region us-east-1
```

2. (Optional) Delete any unwanted tags for the image by specifying the tag of the image you want to delete.

   Note
   When you delete the last tag for an image, the image is deleted.

```
aws ecr-public batch-delete-image \
      --repository-name my-repo \
      --image-ids imageTag=latest \
      --region us-east-1
```

3. Delete the image by specifying the digest of the image to delete.

   Note
   When you delete an image by referencing its digest, the image and all of its tags are deleted.

```
aws ecr-public batch-delete-image \
      --repository-name my-repo \
      --image-ids
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304c7c2c1a9d6fa3e9de6bf552d
      --region us-east-1
```

Container image manifest formats

Amazon ECR Public supports the following container image manifest formats:

- Docker Image Manifest V2 Schema 1 (used with Docker version 1.9 and older)
- Docker Image Manifest V2 Schema 2 (used with Docker version 1.10 and newer)
- Open Container Initiative (OCI) Specifications (v1.0 and up)

Support for Docker Image Manifest V2 Schema 2 provides the following functionality:

- The ability to use multiple tags per image.
- Support for storing Windows container images. For more information, see Pushing Windows Images to Amazon ECR in the *Amazon Elastic Container Service Developer Guide*.

Amazon ECR image manifest conversion

When you push and pull images to and from Amazon ECR Public, your container engine client (for example, Docker) communicates with the public registry to agree on a manifest format that is understood by the client and the registry to use for the image.

When you push an image to Amazon ECR Public with Docker version 1.9 or older, the image manifest format is stored as Docker Image Manifest V2 Schema 1. When you push an image to Amazon ECR Public with Docker version 1.10 or newer, the image manifest format is stored as Docker Image Manifest V2 Schema 2.

When you pull an image from Amazon ECR Public *by tag*, Amazon ECR Public returns the image manifest format that is stored in the repository. The format is returned only if that format is understood by the client. If the stored image manifest format is not understood by the client, Amazon ECR Public converts the image manifest into a format that is understood by the client. For example, if a Docker 1.9 client

requests an image manifest that is stored as Docker Image Manifest V2 Schema 2, Amazon ECR Public returns the manifest in the Docker Image Manifest V2 Schema 1 format. The table below describes the available conversions supported by Amazon ECR Public when an image is pulled *by tag*:

Schema requested by client	Pushed to ECR as V2, schema 1	Pushed to ECR as V2, schema 2	Pushed to ECR as OCI
V2, schema 1	No translation required	Translated to V2, schema 1	Translated to V2, schema 1
V2, schema 2	No translation available, client falls back to V2, schema 1	No translation required	Translated to V2, schema 2
OCI	No translation available	Translated to OCI	No translation required

Important
If you pull an image *by digest*, there is no translation available; your client must understand the image manifest format that is stored in Amazon ECR. If you request a Docker Image Manifest V2 Schema 2 image by digest on a Docker 1.9 or older client, the image pull fails. For more information, see Registry compatibility in the Docker documentation.
In this example, if you request the same image *by tag*, Amazon ECR Public translates the image manifest into a format that the client can understand. The image pull succeeds.

Security in Amazon Elastic Container Registry

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon ECR, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon ECR. The following topics show you how to configure Amazon ECR to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon ECR resources.

Topics

Identity and Access Management for Amazon ECR

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon ECR Public resources. IAM is an AWS service that you can use with no additional charge.

> **Note**
> For IAM information for Amazon ECR private registries, see Identity and Access Management for Amazon Elastic Container Registry in the *Amazon Elastic Container Registry User Guide*.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon ECR Public.

Service user – If you use the Amazon ECR Public service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon ECR Public features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon ECR Public, see Troubleshooting Amazon ECR identity and access (p. 56).

Service administrator – If you're in charge of Amazon ECR Public resources at your company, you probably have full access to Amazon ECR Public. It's your job to determine which Amazon ECR Public features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon ECR Public, see How Amazon ECR Public works with IAM (p. 46).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon ECR Public. To view example Amazon ECR Public identity-based policies that you can use in IAM, see Amazon ECR identity-based policy examples (p. 52).

Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *AWS Account Management Reference Guide*.

IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see test in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. By default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the

resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon ECR Public works with IAM

Before you use IAM to manage access to Amazon ECR, you should understand what IAM features are available to use with Amazon ECR. To get a high-level view of how Amazon ECR and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

Topics

Amazon ECR identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon ECR supports specific actions, resources,

and condition keys when managing a public registry and the resources in a public registry. The image pull permissions can't be changed because Amazon ECR Public repositories are public accessible. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon ECR Public use the following prefix before the action: `ecr-public:`. For example, to grant someone permission to create a public Amazon ECR repository with the Amazon ECR Public `CreateRepository` API operation, you include the `ecr-public:CreateRepository` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Amazon ECR defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "ecr-public:action1",
      "ecr-public:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "ecr-public:Describe*"
```

To see a list of Amazon ECR actions, see Actions, Resources, and Condition Keys for Amazon ECR Public in the *IAM User Guide*.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

An Amazon ECR public repository resource has the following ARN. You don't use a Region name in the ARN.

```
arn:${Partition}:ecr-public::${Account}:repository/${Repository-name}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the `my-repo` public repository in your statement, use the following ARN:

```
"Resource": "arn:aws:ecr-public::123456789012:repository/my-repo"
```

To specify all public repositories that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:ecr-public::123456789012:repository/*"
```

To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "resource1",
      "resource2"
```

To see a list of Amazon ECR Public resource types and their ARNs, see Resources defined by Amazon ECR Public in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon ECR.

Condition Keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

Amazon ECR supports using some global condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Amazon ECR Public condition keys, see Condition keys defined by Amazon ECR Public in the *IAM User Guide*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon ECR.

Examples

To view examples of Amazon ECR identity-based policies, see Amazon ECR identity-based policy examples (p. 52).

Amazon ECR resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on an Amazon ECR Public resource and under what conditions. Amazon ECR supports resource-based permissions policies for Amazon ECR public repositories. Resource-based policies let you grant usage permission to other accounts on a per-resource basis. You can also use a resource-based policy to allow an AWS service to access your Amazon ECR public repositories.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

The Amazon ECR Public service supports only one type of resource-based policy called a *repository policy*, which is attached to a *repository*. This policy defines which principal entities (accounts, users, roles, and federated users) can perform actions on the repository.

To learn how to attach a resource-based policy to a repository, see Public repository policies (p. 27).

Examples

To view examples of Amazon ECR resource-based policies, see Public repository policy examples (p. 30).

Amazon ECR IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Using temporary credentials with Amazon ECR

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon ECR Public supports using temporary credentials.

Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Amazon ECR Public does not support service-linked roles.

AWS managed policies for Amazon ECR Public

Amazon ECR Public provides several managed policies that you can attach to users or Amazon EC2 instances. These policies allow for differing levels of control over Amazon ECR resources and API operations. You can apply these policies directly or use them as starting points for creating your own policies. For more information about each API operation that's mentioned in these policies, see Actions in the *Amazon ECR Public API Reference*.

Topics

AmazonElasticContainerRegistryPublicFullAccess

You can attach the `AmazonElasticContainerRegistryPublicFullAccess` policy to your IAM identities.

This managed policy is a starting point for providing an IAM user or role with full administrator access to manage their use of Amazon ECR Public.

Permissions details

This policy includes the following permissions:

- `ecr-public` – Provides IAM principals full access to all Amazon ECR APIs.
- `sts` – Allows IAM principals to acquire a AWS Security Token Service bearer token for an AWS root user, IAM role, or an IAM user.

The `AmazonElasticContainerRegistryPublicFullAccess` policy is as follows.

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

AmazonElasticContainerRegistryPublicPowerUser

You can attach the `AmazonEC2ContainerRegistryPowerUser` policy to your IAM identities.

This policy grants administratice permissions that allow power user access to Amazon ECR Public. This provides write access to public repositories, but it doesn't allow users to delete public repositories or change the policy documents that are applied to them.

Permissions details

This policy includes the following permissions:

- `ecr-public` – Allows IAM principals to read and write to respositores and read lifecycle policies. IAM principals aren't granted permission to delete repositories or change the lifecycle policies that are applied to them.
- `sts` – Allows IAM principals to acquire a AWS Security Token Service bearer token for an AWS root user, IAM role, or an IAM user.

The `AmazonElasticContainerRegistryPublicPowerUser` policy is as follows.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr-public:GetAuthorizationToken",
                "sts:GetServiceBearerToken",
                "ecr-public:BatchCheckLayerAvailability",
                "ecr-public:GetRepositoryPolicy",
                "ecr-public:DescribeRepositories",
                "ecr-public:DescribeRegistries",
                "ecr-public:DescribeImages",
                "ecr-public:DescribeImageTags",
                "ecr-public:GetRepositoryCatalogData",
                "ecr-public:GetRegistryCatalogData",
                "ecr-public:InitiateLayerUpload",
                "ecr-public:UploadLayerPart",
                "ecr-public:CompleteLayerUpload",
                "ecr-public:PutImage"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonElasticContainerRegistryPublicReadOnly

You can attach the `AmazonElasticContainerRegistryPublicReadOnly` policy to your IAM identities.

This policy grants read-only permissions to Amazon ECR Public. These permissions include the ability to describe public registries, to list and describe public repositories, to describe images within a public repository, and to pull images from Amazon ECR Public with the Docker CLI.

Permissions details

This policy includes the following permissions:

- `ecr` – Allows IAM principals to read repositories and their respective lifecycle policies.
- `sts` – Allows IAM principals to acquire a AWS Security Token Service bearer token for an AWS root user, IAM role, or an IAM user.

The `AmazonElasticContainerRegistryPublicReadOnly` policy is as follows.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "ecr-public:GetAuthorizationToken",
            "sts:GetServiceBearerToken",
            "ecr-public:BatchCheckLayerAvailability",
            "ecr-public:GetRepositoryPolicy",
            "ecr-public:DescribeRepositories",
            "ecr-public:DescribeRegistries",
            "ecr-public:DescribeImages",
            "ecr-public:DescribeImageTags",
            "ecr-public:GetRepositoryCatalogData",
            "ecr-public:GetRegistryCatalogData"
```

```
        ],
        "Resource": "*"
    }]
}
```

Amazon ECR Public updates to AWS managed policies

View details about updates to AWS managed policies for Amazon ECR Public since the time that this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon ECR Public Document history page.

Change	Description	Date
Amazon ECR started tracking changes	Amazon ECR started tracking changes for AWS managed policies.	June 24, 2021
AmazonElasticContainerRegistryPublicReadOnly (p. 51) – New policy	Amazon ECR added a new policy that grants read-only permissions to Amazon ECR Public. These permissions include the ability to describe public registries, to list and describe public repositories, to describe images within a public repository and to pull images from Amazon ECR Public with the Docker CLI.	December 1, 2020
AmazonElasticContainerRegistryPublicPowerUser (p. 50) – New policy	Amazon ECR added a new policy that grants administrative permissions to Amazon ECR Public that allow write access to public repositories. However, these permissions don't allow users to delete public repositories or change the policy documents that are applied to them.	December 1, 2020
AmazonElasticContainerRegistryPublicFullAccess (p. 50) – New policy	Amazon ECR added a new policy that grants full access to Amazon ECR Public.	December 1, 2020

Amazon ECR identity-based policy examples

By default, users and roles don't have permission to create or modify Amazon ECR Public resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies for users that require them.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Amazon ECR, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Elastic Container Registry in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon ECR Public resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or root users in your account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Amazon ECR console

> **Note**
> These permissions are only for Amazon ECR Public resources. For permissions related to your private Amazon ECR resources, see Amazon ECR identity-based policy examples in the *Amazon Elastic Container Registry User Guide*.

To access the Amazon ECR console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon ECR Public resources in your AWS account. If

you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

To ensure that those entities can still use the Amazon ECR console, add the `AmazonElasticContainerRegistryPublicReadOnly` AWS managed policy to the entities. For more information, see Adding Permissions to a User in the *IAM User Guide*:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "ecr-public:GetAuthorizationToken",
            "sts:GetServiceBearerToken",
            "ecr-public:BatchCheckLayerAvailability",
            "ecr-public:GetRepositoryPolicy",
            "ecr-public:DescribeRepositories",
            "ecr-public:DescribeRegistries",
            "ecr-public:DescribeImages",
            "ecr-public:DescribeImageTags",
            "ecr-public:GetRepositoryCatalogData",
            "ecr-public:GetRegistryCatalogData"
        ],
        "Resource": "*"
    }]
}
```

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
```

```
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Accessing an Amazon ECR public repository

In this example, you want to grant an IAM user in your AWS account access to one of your Amazon ECR public repositories, `my-repo`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"GetAuthorizationToken",
         "Effect":"Allow",
         "Action":[
            "ecr-public:GetAuthorizationToken"
         ],
         "Resource":"*"
      },
      {
         "Sid":"ManageRepositoryContents",
         "Effect":"Allow",
         "Action":[
                "ecr-public:BatchCheckLayerAvailability",
                "ecr-public:GetRepositoryPolicy",
                "ecr-public:DescribeRepositories",
                "ecr-public:DescribeImages",
                "ecr-public:InitiateLayerUpload",
                "ecr-public:UploadLayerPart",
                "ecr-public:CompleteLayerUpload",
                "ecr-public:PutImage"
         ],
         "Resource":"arn:aws:ecr-public::123456789012:repository/my-repo"
      }
   ]
}
```

Using tag-based access control

The Amazon ECR Public `CreateRepository` API action enables you to specify tags when you create the repository. For more information, see Tagging an Amazon ECR Public repository (p. 31).

To enable users to tag repositories on creation, they must have permissions to use the action that creates the resource (for example, `ecr-public:CreateRepository`). If tags are specified in the resource-creating action, AWS performs additional authorization on the `ecr-public:CreateRepository` action to verify if users have permissions to create tags.

You can used tag-based access control through IAM policies. The following are examples.

The following policy would only allow an IAM user to create or tag a public repository where the tag key is `environment` and tag value is `dev`.

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

```
            "Sid": "AllowOnlyTagWithVals",
            "Effect": "Allow",
            "Action": [
                "ecr-public:CreateRepository",
                "ecr-public:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": [
                        "dev"
                    ]
                }
            }
        }
    ]
}
```

The following policy would allow an IAM user access to all public repositories unless they were tagged as `key=environment,value=prod`.

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

Troubleshooting Amazon ECR identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon ECR and IAM.

Topics

I am not authorized to perform an action in Amazon ECR Public

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `ecr:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecr:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `ecr:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon ECR Public.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon ECR Public. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm an administrator and want to allow others to access Amazon ECR Public

To allow others to access Amazon ECR Public, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon ECR Public.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my Amazon ECR Public resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon ECR Public supports these features, see How Amazon ECR Public works with IAM (p. 46).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.

- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Logging Amazon ECR Public actions with AWS CloudTrail

Amazon ECR Public is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, a role, or an AWS service in Amazon ECR Public. When activity occurs in Amazon ECR Public, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. Because Amazon ECR Public is a global service, events for the service are logged in **US East (N. Virginia)**.

When a trail is created, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon ECR Public. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using this information, you can determine the request that was made to Amazon ECR Public, the originating IP address, who made the request, when it was made, and additional details.

For more information, see the AWS CloudTrail User Guide.

Amazon ECR Public information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon ECR Public, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon ECR Public, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. When you create a trail in the console, you can apply the trail to a single Region or to all Regions. The trail logs events in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Creating a trail for your AWS account
- AWS service integrations with CloudTrail logs
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon ECR Public API actions are logged by CloudTrail and are documented in the Amazon Elastic Container Registry API Reference. When you perform common tasks, sections are generated in the CloudTrail log files for each API action that is part of that task. For example, when you create a repository, `GetAuthorizationToken` and `CreateRepository` sections are generated in the CloudTrail log files. When you push an image to a repository, `InitiateLayerUpload`, `UploadLayerPart`, `CompleteLayerUpload`, and `PutImage` sections are generated. For examples of these common tasks, see CloudTrail log entry examples (p. 60).

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials

- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail `userIdentity` element.

Understanding Amazon ECR Public log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and other information. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

> **Important**
> Because Amazon ECR Public is a global service, events for the service are logged in **US East (N. Virginia)**.

CloudTrail log entry examples

The following are CloudTrail log entry examples for a few common Amazon ECR tasks.

> **Note**
> These examples have been formatted for improved readability. In a CloudTrail log file, all entries and events are concatenated into a single line. In addition, this example has been limited to a single Amazon ECR Public entry. In a real CloudTrail log file, you see entries and events from multiple AWS services.

Topics

Example: Create repository action

The following example shows a CloudTrail log entry that demonstrates the `CreateRepository` action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKC6C2EXAMPLE:account_name",
        "arn": "arn:aws:iam::123456789012:user/admin",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "admin"
    },
    "eventTime": "2020-11-27T21:51:29Z",
    "eventSource": "ecr-public.amazonaws.com",
    "eventName": "CreateRepository",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/2.0.28 Python/3.7.4 Darwin/18.7.0 botocore/2.0.0dev32",
    "requestParameters": {
        "repositoryName": "ecr-tutorial"
    },
    "responseElements": {
        "repository": {
```

```
            "repositoryArn": "arn:aws:ecr-public::123456789012:repository/ecr-tutorial",
            "registryId": "123456789012",
            "repositoryName": "ecr-tutorial",
            "repositoryUri": "public.ecr.aws/j3y4EXAMPLE/ecr-tutorial",
            "createdAt": "Nov 27, 2020, 9:51:29 PM"
        },
        "catalogData": {}
    },
    "requestID": "852d12c4-2495-451c-9fd6-a1fcEXAMPLE",
    "eventID": "28371107-6ffc-44a1-92d9-564EXAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "ARN": "arn:aws:ecr-public::123456789012:repository/ecr-tutorial"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}
```


Example: Image push action

The following example shows a CloudTrail log entry that demonstrates an image push which uses the `PutImage` action.

Note
When pushing an image, you will also see `InitiateLayerUpload`, `UploadLayerPart`, and `CompleteLayerUpload` references in the CloudTrail logs.


```
{
    "eventVersion": "1.04",
    "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKC6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Mary_Major",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-04-15T16:42:14Z"
   }
  }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr-public.amazonaws.com",
 "eventName": "PutImage",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "203.0.113.12",
 "userAgent": "console.amazonaws.com",
 "requestParameters": {
  "repositoryName": "testrepo",
  "imageTag": "latest",
  "registryId": "123456789012",
  "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
```

```
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest
\": \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 37720774,\n         \"digest
\": \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
  ]\n}"
 },
 "responseElements": {
  "image": {
   "repositoryName": "testrepo",
   "imageManifest": "{\n   \"schemaVersion\": 2,\n   \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n   \"config\": {\n      \"mediaType\":
 \"application/vnd.docker.container.image.v1+json\",\n      \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n   },\n   \"layers\": [\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 846,\n         \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 615,\n         \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 850,\n         \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 168,\n         \"digest
\": \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 37720774,\n         \"digest
\": \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 30432107,\n
```

```
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 197,\n         \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 154,\n         \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 176,\n         \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 183,\n         \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n         \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n         \"size\": 212,\n         \"digest\":
 \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n      }\n
  ]\n}",
   "registryId": "123456789012",
   "imageId": {
    "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
    "imageTag": "latest"
   }
  }
 },
 "requestID": "cf044b7d-5f9d-11e9-9b2a-95983139cc57",
 "eventID": "2bfd4ee2-2178-4a82-a27d-b12939923f0f",
 "resources": [{
  "ARN": "arn:aws:ecr-public:us-east-1:123456789012:repository/testrepo",
  "accountId": "123456789012"
 }],
 "eventType": "AwsApiCall",
 "recipientAccountId": "123456789012"
}
```

Amazon ECR Public service quotas

The following table provides the default service quotas for Amazon ECR Public.

Name	Default	Adjust	Description
Images per repository	Each supported Region: 10,000	Yes	The maximum number of images per repository.
Layer parts	Each supported Region: 1,000	No	The maximum number of layer parts. This is only applicable if you are using Amazon ECR API actions directly to initiate multipart uploads for image push operations.
Maximum layer part size	Each supported Region: 10	No	The maximum size (MiB) of a layer part. This is only applicable if you are using Amazon ECR API actions directly to initiate multipart uploads for image push operations.
Maximum layer size	Each supported Region: 10,000	No	The maximum size per layer.
Minimum layer part size	Each supported Region: 5	No	The minimum size (MiB) of a layer part. This is only applicable if you are using Amazon ECR API actions directly to initiate multipart uploads for image push operations.
Rate of BatchCheckLayerAvailability requests	Each supported Region: 200 per second	Yes	The maximum number of BatchCheckLayerAvailability requests that you can make per second in the current Region. When an image is pushed to a repository, each image layer is checked to verify if it has been uploaded before. If it has been uploaded, then the image layer is skipped.
Rate of CompleteLayerUpload requests	Each supported Region: 10 per second	Yes	The maximum number of CompleteLayerUpload requests that you can make per second in the current Region. When an image is pushed, the CompleteLayerUpload API

Name	Default	Adjust	Description
			is called once per each new image layer to verify that the upload has completed.
Rate of GetAuthorizationToken requests	Each supported Region: 200 per second	Yes	The maximum number of GetAuthorizationToken requests that you can make per second in the current Region.
Rate of InitiateLayerUpload requests	Each supported Region: 200 per second	Yes	The maximum number of InitiateLayerUpload requests that you can make per second in the current Region. When an image is pushed, the InitiateLayerUpload API is called once per image layer that has not already been uploaded. Whether or not an image layer has been uploaded is determined by the BatchCheckLayerAvailability API action.
Rate of PutImage requests	Each supported Region: 10 per second	Yes	The maximum number of PutImage requests that you can make per second in the current Region. When an image is pushed and all new image layers have been uploaded, the PutImage API is called once to create or update the image manifest and the tags associated with the image.
Rate of UploadLayerPart requests	Each supported Region: 260 per second	Yes	The maximum number of UploadLayerPart requests that you can make per second in the current Region. When an image is pushed, each new image layer is uploaded in parts and the UploadLayerPart API is called once per each new image layer part.
Rate of authenticated image pulls	Each supported Region: 10 per second	Yes	The maximum number of authenticated image pulls per second.

Name	Default	Adjust	Description
Rate of image pulls to AWS resources	Each supported Region: 10 per second	No	The maximum number of image pulls per second to resources running on Amazon ECS, Fargate, or Amazon EC2.
Rate of unauthenticated image pulls	Each supported Region: 1 per second	No	The maximum number of unauthenticated image pulls per second.
Registered repositories	Each supported Region: 10,000	Yes	The maximum number of repositories that you can create in this account in the current Region.
Tags per image	Each supported Region: 1,000	No	The maximum number of tags per image.

Managing your Amazon ECR Public service quotas in the AWS Management Console

Amazon ECR Public has integrated with Service Quotas, an AWS service that enables you to view and manage your quotas from a central location. For more information, see What Is Service Quotas? in the *Service Quotas User Guide*.

Service Quotas makes it easy to look up the value of all Amazon ECR service quotas.

To view Amazon ECR service quotas (AWS Management Console)

1. Open the Service Quotas console at https://console.aws.amazon.com/servicequotas/.
2. In the navigation pane, choose **AWS services**.
3. From the **AWS services** list, search for and select **Amazon Elastic Container Registry Public (Amazon ECR Public)**.

   In the **Service quotas** list, you can see the service quota name, applied value (if it is available), AWS default quota, and whether the quota value is adjustable.
4. To view additional information about a service quota, such as the description, choose the quota name.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*.

Amazon ECR Public troubleshooting

This chapter helps you find diagnostic information for Amazon ECR Public, and provides troubleshooting steps for common issues and error messages.

Topics

Authentication issues

Issue: When performing an unauthenticated pull from an Amazon ECR Public repository, you receive an authentication token expired response. This is likely due to the fact that you've previously requested an authentication token from Amazon ECR Public and that token has expired. When the new Amazon ECR Public image pull is performed, the expired token is used and the error is received. The following is an example error.

```
Error response from daemon: pull access denied for
 public.ecr.aws/registry_alias/repository_name, repository does not exist or may require
 'docker login': denied: Your authorization token has expired. Reauthenticate and try
 again.
```

Resolution: To resolve this issue, you can either re-authenticate to Amazon ECR Public or you can log your Docker CLI out of the Amazon ECR Public registry and re-attempt your unauthenticated image pull.

```
docker logout public.ecr.aws
```

Document history

The following table describes the important changes to the documentation since the last release of Amazon ECR Public. We also update the documentation frequently to address the feedback that you send us.

Change	Description	Date
AWS Managed Policies for Amazon ECR Public	Amazon ECR Public added documentation of AWS managed policies. For more information, see AWS managed policies for Amazon ECR Public (p. 49).	24 June 2021
Tagging support	Added support for tagging your public repositories. For more information, see Tagging an Amazon ECR Public repository (p. 31).	24 Feb 2021
Amazon ECR Public general availability	Amazon Elastic Container Registry Public is a public AWS container image registry service that is secure, scalable, and reliable.	1 Dec 2020

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.