# Amazon EC2 Container Registry

## API Reference

## API Version 2015-09-21

# Amazon EC2 Container Registry: API Reference

# Table of Contents

# Welcome

Amazon EC2 Container Registry (Amazon ECR) is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images. Amazon ECR provides a secure, scalable, and reliable registry. Amazon ECR supports private Docker repositories with resource-based permissions using AWS IAM so that specific users or Amazon EC2 instances can access repositories and images. Developers can use the Docker CLI to author and manage images.

This document was last published on December 9, 2016.

# Actions

The following actions are supported:

# BatchCheckLayerAvailability

Check the availability of multiple image layers in a specified registry and repository.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "layerDigests": [ "string" ],
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**layerDigests (p. 3)**
The digests of the image layers to check.
Type: array of Strings
Array Members: Minimum number of 1 item. Maximum number of 100 items.
Length Constraints: Minimum length of 0. Maximum length of 1000.
Required: Yes

**registryId (p. 3)**
The AWS account ID associated with the registry that contains the image layers to check. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 3)**
The name of the repository that is associated with the image layers to check.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

## Response Syntax

```
{
   "failures": [
      {
         "failureCode": "string",
         "failureReason": "string",
         "layerDigest": "string"
      }
   ],
```

```
   "layers": [
      {
         "layerAvailability": "string",
         "layerDigest": "string",
         "layerSize": number
      }
   ]
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**failures (p. 3)**

Any failures associated with the call.

Type: array of LayerFailure (p. 60) objects

**layers (p. 3)**

A list of image layer objects corresponding to the image layer references in the request.

Type: array of Layer (p. 59) objects

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example checks the availability of an image layer in the `ubuntu` repository.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
```

```
Accept-Encoding: identity
Content-Length: 121
X-Amz-Target:
 AmazonEC2ContainerRegistry_V20150921.BatchCheckLayerAvailability
X-Amz-Date: 20151214T224018Z
User-Agent: aws-cli/1.9.10 Python/2.7.10 Darwin/14.5.0 botocore/1.3.10
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "layerDigests": [
    "sha256:622b8fa00f815b90439859cc2202b493383761bd3eaf942462282ba1f2dbf5a5"
  ],
  "repositoryName": "ubuntu"
}
```

## Sample Response

```
{
  "failures": [],
  "layers": [
    {
      "layerAvailability": "AVAILABLE",
      "layerDigest":
 "sha256:622b8fa00f815b90439859cc2202b493383761bd3eaf942462282ba1f2dbf5a5",
      "layerSize": 680
    }
  ]
}
```

# BatchDeleteImage

Deletes a list of specified images within a specified repository. Images are specified with either `imageTag` or `imageDigest`.

## Request Syntax

```
{
   "imageIds": [
      {
         "imageDigest": "string",
         "imageTag": "string"
      }
   ],
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**imageIds (p. 6)**

A list of image ID references that correspond to images to delete. The format of the `imageIds` reference is `imageTag=tag` or `imageDigest=digest`.

Type: array of ImageIdentifier (p. 58) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Required: Yes

**registryId (p. 6)**

The AWS account ID associated with the registry that contains the image to delete. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 6)**

The repository that contains the image to delete.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "failures": [
      {
         "failureCode": "string",
         "failureReason": "string",
         "imageId": {
```

```
            "imageDigest": "string",
            "imageTag": "string"
         }
      }
   ],
   "imageIds": [
      {
         "imageDigest": "string",
         "imageTag": "string"
      }
   ]
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**failures (p. 6)**

Any failures associated with the call.

Type: array of ImageFailure (p. 57) objects

**imageIds (p. 6)**

The image IDs of the deleted images.

Type: array of ImageIdentifier (p. 58) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example deletes an image in the `ubuntu` repository with the `imageTag` value of `precise`.

## Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 67
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.BatchDeleteImage
X-Amz-Date: 20151201T214807Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "ubuntu",
  "imageIds": [
    {
      "imageTag": "precise"
    }
  ]
}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 01 Dec 2015 21:48:08 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 139
Connection: keep-alive
x-amzn-RequestId: 355645b0-9875-11e5-9018-5fa50b14e31d

{
  "failures": [],
  "imageIds": [
    {
      "imageDigest":
 "sha256:19665f1e6d1e504117a1743c0a3d3753086354a38375961f2e665416ef4b1b2f",
      "imageTag": "precise"
    }
  ]
}
```

# BatchGetImage

Gets detailed information for specified images within a specified repository. Images are specified with either `imageTag` or `imageDigest`.

## Request Syntax

```
{
   "imageIds": [
      {
         "imageDigest": "string",
         "imageTag": "string"
      }
   ],
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**imageIds (p. 9)**

A list of image ID references that correspond to images to describe. The format of the `imageIds` reference is `imageTag=tag` or `imageDigest=digest`.

Type: array of ImageIdentifier (p. 58) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Required: Yes

**registryId (p. 9)**

The AWS account ID associated with the registry that contains the images to describe. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 9)**

The repository that contains the images to describe.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "failures": [
      {
         "failureCode": "string",
         "failureReason": "string",
         "imageId": {
```

```
            "imageDigest": "string",
            "imageTag": "string"
         }
      }
   ],
   "images": [
      {
         "imageId": {
            "imageDigest": "string",
            "imageTag": "string"
         },
         "imageManifest": "string",
         "registryId": "string",
         "repositoryName": "string"
      }
   ]
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**failures (p. 9)**
: Any failures associated with the call.

  Type: array of ImageFailure (p. 57) objects

**images (p. 9)**
: A list of image objects corresponding to the image references in the request.

  Type: array of Image (p. 55) objects

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**
: The specified parameter is invalid. Review the available parameters for the API request.

  HTTP Status Code: 400

**RepositoryNotFoundException**
: The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

  HTTP Status Code: 400

**ServerException**
: These errors are usually caused by a server-side issue.

  HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

# Example

This example describes an image in the `ubuntu` repository with the `imageTag` value of `precise`.

## Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 67
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.BatchGetImage
X-Amz-Date: 20151201T211740Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "ubuntu",
  "imageIds": [
    {
      "imageTag": "precise"
    }
  ]
}
```

# CompleteLayerUpload

Inform Amazon ECR that the image layer upload for a specified registry, repository name, and upload ID, has completed. You can optionally provide a `sha256` digest of the image layer for data validation purposes.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "layerDigests": [ "string" ],
   "registryId": "string",
   "repositoryName": "string",
   "uploadId": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**layerDigests (p. 12)**

The `sha256` digest of the image layer.

Type: array of Strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Pattern: `[a-zA-Z0-9-_+.]+:[a-fA-F0-9]+`

Required: Yes

**registryId (p. 12)**

The AWS account ID associated with the registry to which to upload layers. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 12)**

The name of the repository to associate with the image layer.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

**uploadId (p. 12)**

The upload ID from a previous InitiateLayerUpload (p. 39) operation to associate with the image layer.

Type: String

Pattern: `[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}`

Required: Yes

# Response Syntax

```
{
   "layerDigest": "string",
   "registryId": "string",
   "repositoryName": "string",
   "uploadId": "string"
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**layerDigest (p. 13)**

The `sha256` digest of the image layer.

Type: String

Pattern: `[a-zA-Z0-9-_+.]+:[a-fA-F0-9]+`

**registryId (p. 13)**

The registry ID associated with the request.

Type: String

Pattern: `[0-9]{12}`

**repositoryName (p. 13)**

The repository name associated with the request.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

**uploadId (p. 13)**

The upload ID associated with the layer.

Type: String

Pattern: `[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**EmptyUploadException**

The specified layer upload does not contain any layer parts.

HTTP Status Code: 400

**InvalidLayerException**

The layer digest calculation performed by Amazon ECR upon receipt of the image layer does not match the digest specified.

HTTP Status Code: 400

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**LayerAlreadyExistsException**

The image layer already exists in the associated repository.

HTTP Status Code: 400

**LayerPartTooSmallException**

Layer parts must be at least 5 MiB in size.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

**UploadNotFoundException**

The upload could not be found, or the specified upload id is not valid for this repository.

HTTP Status Code: 400

# CreateRepository

Creates an image repository.

## Request Syntax

```
{
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**repositoryName (p. 15)**

The name to use for the repository. The repository name may be specified on its own (such as `nginx-web-app`) or it can be prepended with a namespace to group the repository into a category (such as `project-a/nginx-web-app`).

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "repository": {
      "createdAt": number,
      "registryId": "string",
      "repositoryArn": "string",
      "repositoryName": "string",
      "repositoryUri": "string"
   }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**repository (p. 15)**

The repository that was created.

Type: Repository (p. 62) object

## Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**LimitExceededException**

The operation did not succeed because it would have exceeded a service limit for your account. For more information, see Amazon ECR Default Service Limits in the Amazon EC2 Container Registry User Guide.

HTTP Status Code: 400

**RepositoryAlreadyExistsException**

The specified repository already exists in the specified registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example creates a repository called `nginx-web-app` inside the `project-a` namespace in the default registry for an account.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 45
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.CreateRepository
X-Amz-Date: 20151130T203458Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "project-a/nginx-web-app"
}
```

### Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 30 Nov 2015 20:34:58 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 175
Connection: keep-alive
x-amzn-RequestId: 123a4b56-7c89-01d2-3ef4-example5678f

{
```

```
  "repository": {
    "registryId": "012345678910",
    "repositoryArn": "arn:aws:ecr:us-east-1:012345678910:repository/project-
a/nginx-web-app",
    "repositoryName": "project-a/nginx-web-app"
  }
}
```

# DeleteRepository

Deletes an existing image repository. If a repository contains images, you must use the `force` option to delete it.

## Request Syntax

```
{
   "force": boolean,
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**force (p. 18)**
Force the deletion of the repository if it contains images.
Type: Boolean
Required: No

**registryId (p. 18)**
The AWS account ID associated with the registry that contains the repository to delete. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 18)**
The name of the repository to delete.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

## Response Syntax

```
{
   "repository": {
      "createdAt": number,
      "registryId": "string",
      "repositoryArn": "string",
      "repositoryName": "string",
      "repositoryUri": "string"
   }
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**repository (p. 18)**
The repository that was deleted.
Type: Repository (p. 62) object

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**RepositoryNotEmptyException**
The specified repository contains images. To delete a repository that contains images, you must force the deletion with the `force` parameter.
HTTP Status Code: 400

**RepositoryNotFoundException**
The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example deletes a repository named `ubuntu` in the default registry for an account.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 43
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.DeleteRepository
X-Amz-Date: 20151130T201424Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "ubuntu",
  "force": true
```

```
}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 30 Nov 2015 20:14:24 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 141
Connection: keep-alive
x-amzn-RequestId: 123a4b56-7c89-01d2-3ef4-example5678f

{
  "repository": {
    "registryId": "012345678910",
    "repositoryArn": "arn:aws:ecr:us-east-1:012345678910:repository/ubuntu",
    "repositoryName": "ubuntu"
  }
}
```

# DeleteRepositoryPolicy

Deletes the repository policy from a specified repository.

## Request Syntax

```
{
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**registryId (p. 21)**

The AWS account ID associated with the registry that contains the repository policy to delete. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 21)**

The name of the repository that is associated with the repository policy to delete.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "policyText": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**policyText (p. 21)**

The JSON repository policy that was deleted from the repository.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 10240.

**registryId (p. 21)**

The registry ID associated with the request.

Type: String

Pattern: `[0-9]{12}`

**repositoryName (p. 21)**

The repository name associated with the request.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**RepositoryPolicyNotFoundException**

The specified repository and registry combination does not have an associated repository policy.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example deletes the repository policy from the `ubuntu` repository.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 28
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.DeleteRepositoryPolicy
X-Amz-Date: 20151215T003722Z
User-Agent: aws-cli/1.9.10 Python/2.7.10 Darwin/14.5.0 botocore/1.3.10
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "ubuntu"
```

```
}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 15 Dec 2015 00:37:22 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 301
Connection: keep-alive
x-amzn-RequestId: 01817918-a2c4-11e5-a19f-014c7a9aad99

{
  "policyText": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"AllowPull\",\n    \"Effect\" : \"Allow\",\n
  \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchGetImage\",
 \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}",
  "registryId": "012345678910",
  "repositoryName": "ubuntu"
}
```

# DescribeImages

Returns metadata about the images in a repository, including image size and creation date.

**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, so it may return a larger image size than the image sizes returned by DescribeImages (p. 24).

## Request Syntax

```
{
   "filter": {
      "tagStatus": "string"
   },
   "imageIds": [
      {
         "imageDigest": "string",
         "imageTag": "string"
      }
   ],
   "maxResults": number,
   "nextToken": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**filter (p. 24)**
The filter key and value with which to filter your `DescribeImages` results.
Type: DescribeImagesFilter (p. 54) object
Required: No

**imageIds (p. 24)**
The list of image IDs for the requested repository.
Type: array of ImageIdentifier (p. 58) objects
Array Members: Minimum number of 1 item. Maximum number of 100 items.
Required: No

**maxResults (p. 24)**
The maximum number of repository results returned by `DescribeImages` in paginated output. When this parameter is used, `DescribeImages` only returns `maxResults` results in a single page along with a `nextToken` response element. The remaining results of the initial request can be seen by sending another `DescribeImages` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `DescribeImages` returns up to 100 results and a `nextToken` value, if applicable.
Type: Integer
Valid Range: Minimum value of 1. Maximum value of 100.
Required: No

**nextToken (p. 24)**

The `nextToken` value returned from a previous paginated `DescribeImages` request where `maxResults` was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the `nextToken` value. This value is `null` when there are no more results to return.

Type: String

Required: No

**registryId (p. 24)**

The AWS account ID associated with the registry that contains the repository in which to describe images. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 24)**

A list of repositories to describe. If this parameter is omitted, then all repositories in a registry are described.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

# Response Syntax

```
{
   "imageDetails": [
      {
         "imageDigest": "string",
         "imagePushedAt": number,
         "imageSizeInBytes": number,
         "imageTags": [ "string" ],
         "registryId": "string",
         "repositoryName": "string"
      }
   ],
   "nextToken": "string"
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**imageDetails (p. 25)**

A list of ImageDetail (p. 56) objects that contain data about the image.

Type: array of ImageDetail (p. 56) objects

**nextToken (p. 25)**

The `nextToken` value to include in a future `DescribeImages` request. When the results of a `DescribeImages` request exceed `maxResults`, this value can be used to retrieve the next page of results. This value is `null` when there are no more results to return.

Type: String

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**ImageNotFoundException**
The image requested does not exist in the specified repository.
HTTP Status Code: 400

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**RepositoryNotFoundException**
The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example describes the images in a repository named project-a in the default account.

### Sample Request

```
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 31
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.DescribeImages
X-Amz-Date: 20160927T225311Z
User-Agent: aws-cli/1.10.21 Python/2.7.9 Windows/2008ServerR2 botocore/1.4.12
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{"repositoryName": "project-a"}
```

### Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 27 Sep 2016 22:53:13 GMT
Content-Type: application/x-amz-json-1.1
```

```
Content-Length: 247
Connection: keep-alive
x-amzn-RequestId: 2b3f2d31-8505-11e6-8d5b-3b0f4fc13a96

{
  "imageDetails": [
    {
      "imageDigest":
 "sha256:b18d7d610dbe213fab3dbc448e0cfd36a51237daa27d9168bf6b19ab21b090d1",
      "imagePushedAt": 1475016331,
      "imageSizeInBytes": 210761064,
      "imageTags": [
        "latest"
      ],
      "registryId": "012345678901",
      "repositoryName": "project-a"
    }
  ]
}
```

# DescribeRepositories

Describes image repositories in a registry.

## Request Syntax

```
{
   "maxResults": number,
   "nextToken": "string",
   "registryId": "string",
   "repositoryNames": [ "string" ]
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**maxResults (p. 28)**

The maximum number of repository results returned by `DescribeRepositories` in paginated output. When this parameter is used, `DescribeRepositories` only returns `maxResults` results in a single page along with a `nextToken` response element. The remaining results of the initial request can be seen by sending another `DescribeRepositories` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `DescribeRepositories` returns up to 100 results and a `nextToken` value, if applicable.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 28)**

The `nextToken` value returned from a previous paginated `DescribeRepositories` request where `maxResults` was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the `nextToken` value. This value is `null` when there are no more results to return.

> **Note**
> This token should be treated as an opaque identifier that is only used to retrieve the next items in a list and not for other programmatic purposes.

Type: String

Required: No

**registryId (p. 28)**

The AWS account ID associated with the registry that contains the repositories to be described. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryNames (p. 28)**

A list of repositories to describe. If this parameter is omitted, then all repositories in a registry are described.

Type: array of Strings

Array Members: Minimum number of 1 item. Maximum number of 100 items.

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: No

# Response Syntax

```
{
   "nextToken": "string",
   "repositories": [
      {
         "createdAt": number,
         "registryId": "string",
         "repositoryArn": "string",
         "repositoryName": "string",
         "repositoryUri": "string"
      }
   ]
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**nextToken (p. 29)**

The `nextToken` value to include in a future `DescribeRepositories` request. When the results of a `DescribeRepositories` request exceed `maxResults`, this value can be used to retrieve the next page of results. This value is `null` when there are no more results to return.

Type: String

**repositories (p. 29)**

A list of repository objects corresponding to valid repositories.

Type: array of Repository (p. 62) objects

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

# Example

This example describes the repositories in the default registry for an account.

## Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.DescribeRepositories
X-Amz-Date: 20151201T215906Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 01 Dec 2015 21:59:06 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 268
Connection: keep-alive
x-amzn-RequestId: be290452-9876-11e5-9e99-4dbebf3369b0

{
  "repositories": [
    {
      "registryId": "012345678910",
      "repositoryArn": "arn:aws:ecr:us-east-1:012345678910:repository/
ubuntu",
      "repositoryName": "ubuntu"
    },
    {
      "registryId": "012345678910",
      "repositoryArn": "arn:aws:ecr:us-east-1:012345678910:repository/test",
      "repositoryName": "test"
    }
  ]
}
```

# GetAuthorizationToken

Retrieves a token that is valid for a specified registry for 12 hours. This command allows you to use the `docker` CLI to push and pull images with Amazon ECR. If you do not specify a registry, the default registry is assumed.

The `authorizationToken` returned for each registry specified is a base64 encoded string that can be decoded and used in a `docker login` command to authenticate to a registry. The AWS CLI offers an `aws ecr get-login` command that simplifies the login process.

## Request Syntax

```
{
   "registryIds": [ "string" ]
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**registryIds (p. 31)**

A list of AWS account IDs that are associated with the registries for which to get authorization tokens. If you do not specify a registry, the default registry is assumed.

Type: array of Strings

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Pattern: `[0-9]{12}`

Required: No

## Response Syntax

```
{
   "authorizationData": [
      {
         "authorizationToken": "string",
         "expiresAt": number,
         "proxyEndpoint": "string"
      }
   ]
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**authorizationData (p. 31)**

A list of authorization token data objects that correspond to the `registryIds` values in the request.

Type: array of AuthorizationData (p. 53) objects

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example gets an authorization token for your default registry.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.GetAuthorizationToken
X-Amz-Date: 20151129T221940Z
User-Agent: aws-cli/1.9.9 Python/2.7.10 Darwin/14.5.0 botocore/1.3.9
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{}
```

### Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Sun, 29 Nov 2015 22:19:39 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 1590
Connection: keep-alive
x-amzn-RequestId: 123a4b56-7c89-01d2-3ef4-example5678f

{
  "authorizationData": [
    {
      "authorizationToken": "QVdTOkNpQzErSHF1ZXZPcUR...",
      "expiresAt": 1448878779.809,
      "proxyEndpoint": "https://012345678910.dkr.ecr.us-east-1.amazonaws.com"
    }
```

```
  ]
}
```

# GetDownloadUrlForLayer

Retrieves the pre-signed Amazon S3 download URL corresponding to an image layer. You can only get URLs for image layers that are referenced in an image.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "layerDigest": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**layerDigest (p. 34)**
The digest of the image layer to download.
Type: String
Pattern: `[a-zA-Z0-9-_+.]+:[a-fA-F0-9]+`
Required: Yes

**registryId (p. 34)**
The AWS account ID associated with the registry that contains the image layer to download. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 34)**
The name of the repository that is associated with the image layer to download.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

## Response Syntax

```
{
   "downloadUrl": "string",
   "layerDigest": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**downloadUrl (p. 34)**
The pre-signed Amazon S3 download URL for the requested layer.
Type: String

**layerDigest (p. 34)**
The digest of the image layer to download.
Type: String
Pattern: `[a-zA-Z0-9-_+.]+:[a-fA-F0-9]+`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**LayerInaccessibleException**
The specified layer is not available because it is not associated with an image. Unassociated image layers may be cleaned up at any time.
HTTP Status Code: 400

**LayersNotFoundException**
The specified layers could not be found, or the specified layer is not valid for this repository.
HTTP Status Code: 400

**RepositoryNotFoundException**
The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# GetRepositoryPolicy

Retrieves the repository policy for a specified repository.

## Request Syntax

```
{
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**registryId (p. 36)**

The AWS account ID associated with the registry that contains the repository. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 36)**

The name of the repository whose policy you want to retrieve.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "policyText": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**policyText (p. 36)**

The JSON repository policy text associated with the repository.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 10240.

**registryId (p. 36)**

The registry ID associated with the request.

Type: String

Pattern: `[0-9]{12}`

**repositoryName (p. 36)**

The repository name associated with the request.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**RepositoryPolicyNotFoundException**

The specified repository and registry combination does not have an associated repository policy.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

This example gets the repository policy for the `ubuntu` repository.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
Accept-Encoding: identity
Content-Length: 28
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.GetRepositoryPolicy
X-Amz-Date: 20151215T002404Z
User-Agent: aws-cli/1.9.10 Python/2.7.10 Darwin/14.5.0 botocore/1.3.10
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
  "repositoryName": "ubuntu"
```

```
}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Tue, 15 Dec 2015 00:24:04 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 301
Connection: keep-alive
x-amzn-RequestId: 25da0b72-a2c2-11e5-8543-ebda6fb1393b

{
  "policyText": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"AllowPull\",\n    \"Effect\" : \"Allow\",\n
  \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchGetImage\",
 \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}",
  "registryId": "012345678910",
  "repositoryName": "ubuntu"
}
```

# InitiateLayerUpload

Notify Amazon ECR that you intend to upload an image layer.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**registryId (p. 39)**
The AWS account ID associated with the registry that you intend to upload layers to. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 39)**
The name of the repository that you intend to upload layers to.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

## Response Syntax

```
{
   "partSize": number,
   "uploadId": "string"
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

**partSize (p. 39)**
The size, in bytes, that Amazon ECR expects future layer part uploads to be.
Type: Long
Valid Range: Minimum value of 0.

**uploadId (p. 39)**

The upload ID for the layer upload. This parameter is passed to further UploadLayerPart (p. 49) and CompleteLayerUpload (p. 12) operations.

Type: String

Pattern: `[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# ListImages

Lists all the image IDs for a given repository.

You can filter images based on whether or not they are tagged by setting the `tagStatus` parameter to `TAGGED` or `UNTAGGED`. For example, you can filter your results to return only `UNTAGGED` images and then pipe that result to a BatchDeleteImage (p. 6) operation to delete them. Or, you can filter your results to return only `TAGGED` images to list all of the tags in your repository.

## Request Syntax

```
{
   "filter": {
      "tagStatus": "string"
   },
   "maxResults": number,
   "nextToken": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**filter (p. 41)**

The filter key and value with which to filter your `ListImages` results.

Type: ListImagesFilter (p. 61) object

Required: No

**maxResults (p. 41)**

The maximum number of image results returned by `ListImages` in paginated output. When this parameter is used, `ListImages` only returns `maxResults` results in a single page along with a `nextToken` response element. The remaining results of the initial request can be seen by sending another `ListImages` request with the returned `nextToken` value. This value can be between 1 and 100. If this parameter is not used, then `ListImages` returns up to 100 results and a `nextToken` value, if applicable.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 41)**

The `nextToken` value returned from a previous paginated `ListImages` request where `maxResults` was used and the results exceeded the value of that parameter. Pagination continues from the end of the previous results that returned the `nextToken` value. This value is `null` when there are no more results to return.

> **Note**
> This token should be treated as an opaque identifier that is only used to retrieve the next items in a list and not for other programmatic purposes.

Type: String

Required: No

**registryId (p. 41)**

The AWS account ID associated with the registry that contains the repository to list images in. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 41)**

The repository whose image IDs are to be listed.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

# Response Syntax

```
{
   "imageIds": [
      {
         "imageDigest": "string",
         "imageTag": "string"
      }
   ],
   "nextToken": "string"
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**imageIds (p. 42)**

The list of image IDs for the requested repository.

Type: array of ImageIdentifier (p. 58) objects

Array Members: Minimum number of 1 item. Maximum number of 100 items.

**nextToken (p. 42)**

The `nextToken` value to include in a future `ListImages` request. When the results of a `ListImages` request exceed `maxResults`, this value can be used to retrieve the next page of results. This value is `null` when there are no more results to return.

Type: String

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**

The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

# PutImage

Creates or updates the image manifest associated with an image.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "imageManifest": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**imageManifest (p. 44)**
The image manifest corresponding to the image to be uploaded.
Type: String
Required: Yes

**registryId (p. 44)**
The AWS account ID associated with the registry that contains the repository in which to put the image. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 44)**
The name of the repository in which to put the image.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

## Response Syntax

```
{
   "image": {
      "imageId": {
         "imageDigest": "string",
         "imageTag": "string"
      },
      "imageManifest": "string",
      "registryId": "string",
      "repositoryName": "string"
   }
```

```
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**image (p. 44)**
Details of the image uploaded.
Type: Image (p. 55) object

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**ImageAlreadyExistsException**
The specified image has already been pushed, and there are no changes to the manifest or image tag since the last push.
HTTP Status Code: 400

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**LayersNotFoundException**
The specified layers could not be found, or the specified layer is not valid for this repository.
HTTP Status Code: 400

**LimitExceededException**
The operation did not succeed because it would have exceeded a service limit for your account. For more information, see Amazon ECR Default Service Limits in the Amazon EC2 Container Registry User Guide.
HTTP Status Code: 400

**RepositoryNotFoundException**
The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# SetRepositoryPolicy

Applies a repository policy on a specified repository to control access permissions.

## Request Syntax

```
{
   "force": boolean,
   "policyText": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**force (p. 46)**
If the policy you are attempting to set on a repository policy would prevent you from setting another policy in the future, you must force the SetRepositoryPolicy (p. 46) operation. This is intended to prevent accidental repository lock outs.

Type: Boolean

Required: No

**policyText (p. 46)**
The JSON repository policy text to apply to the repository.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 10240.

Required: Yes

**registryId (p. 46)**
The AWS account ID associated with the registry that contains the repository. If you do not specify a registry, the default registry is assumed.

Type: String

Pattern: `[0-9]{12}`

Required: No

**repositoryName (p. 46)**
The name of the repository to receive the policy.

Type: String

Length Constraints: Minimum length of 2. Maximum length of 256.

Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

Required: Yes

## Response Syntax

```
{
   "policyText": "string",
   "registryId": "string",
   "repositoryName": "string"
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**policyText (p. 46)**
The JSON repository policy text applied to the repository.
Type: String
Length Constraints: Minimum length of 0. Maximum length of 10240.

**registryId (p. 46)**
The registry ID associated with the request.
Type: String
Pattern: `[0-9]{12}`

**repositoryName (p. 46)**
The repository name associated with the request.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.
HTTP Status Code: 400

**RepositoryNotFoundException**
The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.
HTTP Status Code: 400

**ServerException**
These errors are usually caused by a server-side issue.
HTTP Status Code: 500

# Example

In the following example or examples, the Authorization header contents (`AUTHPARAMS`) must be replaced with an AWS Signature Version 4 signature. For more information about creating these signatures, see Signature Version 4 Signing Process in the *AWS General Reference*.

You only need to learn how to sign HTTP requests if you intend to manually create them. When you use the AWS Command Line Interface (AWS CLI) or one of the AWS SDKs to make requests to AWS, these tools automatically sign the requests for you with the access key that you specify when you configure the tools. When you use these tools, you don't need to learn how to sign requests yourself.

## Example

The following example sets a repository policy on the `ubuntu` repository that allows all AWS accounts to pull from it.

### Sample Request

```
POST / HTTP/1.1
Host: ecr.us-east-1.amazonaws.com
```

```
Accept-Encoding: identity
Content-Length: 223
X-Amz-Target: AmazonEC2ContainerRegistry_V20150921.SetRepositoryPolicy
X-Amz-Date: 20151214T235302Z
User-Agent: aws-cli/1.9.10 Python/2.7.10 Darwin/14.5.0 botocore/1.3.10
Content-Type: application/x-amz-json-1.1
Authorization: AWUTHPARAMS

{
  "policyText": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"AllowPull\",\"Effect\":\"Allow\",\"Action\":[\"ecr:BatchGetImage\",
\"ecr:GetDownloadUrlForLayer\"],\"Principal\":\"*\"}]}\n",
  "repositoryName": "ubuntu"
}
```

## Sample Response

```
HTTP/1.1 200 OK
Server: Server
Date: Mon, 14 Dec 2015 23:53:02 GMT
Content-Type: application/x-amz-json-1.1
Content-Length: 301
Connection: keep-alive
x-amzn-RequestId: cfc3ead9-a2bd-11e5-91c7-7126cb670c2b

{
  "policyText": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"AllowPull\",\n    \"Effect\" : \"Allow\",\n
  \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchGetImage\",
 \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}",
  "registryId": "012345678910",
  "repositoryName": "ubuntu"
}
```

# UploadLayerPart

Uploads an image layer part to Amazon ECR.

**Note**
This operation is used by the Amazon ECR proxy, and it is not intended for general use by customers. Use the `docker` CLI to pull, tag, and push images.

## Request Syntax

```
{
   "layerPartBlob": blob,
   "partFirstByte": number,
   "partLastByte": number,
   "registryId": "string",
   "repositoryName": "string",
   "uploadId": "string"
}
```

## Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 63).

The request accepts the following data in JSON format.

**layerPartBlob (p. 49)**
The base64-encoded layer part payload.
Type: Base64-encoded binary data
Required: Yes

**partFirstByte (p. 49)**
The integer value of the first byte of the layer part.
Type: Long
Valid Range: Minimum value of 0.
Required: Yes

**partLastByte (p. 49)**
The integer value of the last byte of the layer part.
Type: Long
Valid Range: Minimum value of 0.
Required: Yes

**registryId (p. 49)**
The AWS account ID associated with the registry that you are uploading layer parts to. If you do not specify a registry, the default registry is assumed.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName (p. 49)**
The name of the repository that you are uploading layer parts to.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: Yes

**uploadId (p. 49)**
The upload ID from a previous InitiateLayerUpload (p. 39) operation to associate with the layer part upload.
Type: String
Pattern: `[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}`
Required: Yes

# Response Syntax

```
{
   "lastByteReceived": number,
   "registryId": "string",
   "repositoryName": "string",
   "uploadId": "string"
}
```

# Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

**lastByteReceived (p. 50)**
The integer value of the last byte received in the request.
Type: Long
Valid Range: Minimum value of 0.

**registryId (p. 50)**
The registry ID associated with the request.
Type: String
Pattern: `[0-9]{12}`

**repositoryName (p. 50)**
The repository name associated with the request.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`

**uploadId (p. 50)**
The upload ID associated with the request.
Type: String
Pattern: `[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}`

# Errors

For information about the errors that are common to all actions, see Common Errors (p. 65).

**InvalidLayerPartException**
The layer part size is not valid, or the first byte specified is not consecutive to the last byte of a previous layer part upload.
HTTP Status Code: 400

**InvalidParameterException**
The specified parameter is invalid. Review the available parameters for the API request.

HTTP Status Code: 400

**LimitExceededException**

The operation did not succeed because it would have exceeded a service limit for your account. For more information, see Amazon ECR Default Service Limits in the Amazon EC2 Container Registry User Guide.

HTTP Status Code: 400

**RepositoryNotFoundException**

The specified repository could not be found. Check the spelling of the specified repository and ensure that you are performing operations on the correct registry.

HTTP Status Code: 400

**ServerException**

These errors are usually caused by a server-side issue.

HTTP Status Code: 500

**UploadNotFoundException**

The upload could not be found, or the specified upload id is not valid for this repository.

HTTP Status Code: 400

# Data Types

The Amazon EC2 Container Registry API contains several data types that various actions use. This section describes each data type in detail.

**Note**
The order of each element in a data type structure is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

# AuthorizationData

An object representing authorization data for an Amazon ECR registry.

## Contents

**authorizationToken**

A base64-encoded string that contains authorization data for the specified Amazon ECR registry. When the string is decoded, it is presented in the format `user:password` for private registry authentication using `docker login`.

Type: String

Pattern: `^\S+$`

Required: No

**expiresAt**

The Unix time in seconds and milliseconds when the authorization token expires. Authorization tokens are valid for 12 hours.

Type: Timestamp

Required: No

**proxyEndpoint**

The registry URL to use for this authorization token in a `docker login` command. The Amazon ECR registry URL format is `https://aws_account_id.dkr.ecr.region.amazonaws.com`. For example, `https://012345678910.dkr.ecr.us-east-1.amazonaws.com`..

Type: String

Required: No

# DescribeImagesFilter

An object representing a filter on a DescribeImages (p. 24) operation.

## Contents

**tagStatus**

The tag status with which to filter your DescribeImages (p. 24) results. You can filter results based on whether they are `TAGGED` or `UNTAGGED`.

Type: String

Valid Values: `TAGGED` | `UNTAGGED`

Required: No

# Image

An object representing an Amazon ECR image.

## Contents

**imageId**
An object containing the image tag and image digest associated with an image.
Type: ImageIdentifier (p. 58) object
Required: No

**imageManifest**
The image manifest associated with the image.
Type: String
Required: No

**registryId**
The AWS account ID associated with the registry containing the image.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName**
The name of the repository associated with the image.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: No

# ImageDetail

An object that describes an image returned by a DescribeImages (p. 24) operation.

## Contents

**imageDigest**
The `sha256` digest of the image manifest.
Type: String
Required: No

**imagePushedAt**
The date and time, expressed in standard JavaScript date format, at which the current image was pushed to the repository.
Type: Timestamp
Required: No

**imageSizeInBytes**
The size, in bytes, of the image in 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, so it may return a larger image size than the image sizes returned by DescribeImages (p. 24).

Type: Long
Required: No

**imageTags**
The list of tags associated with this image.
Type: array of Strings
Required: No

**registryId**
The AWS account ID associated with the registry to which this image belongs.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryName**
The name of the repository to which this image belongs.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: No

# ImageFailure

An object representing an Amazon ECR image failure.

## Contents

**failureCode**
The code associated with the failure.
Type: String
Valid Values: `InvalidImageDigest | InvalidImageTag | ImageTagDoesNotMatchDigest | ImageNotFound | MissingDigestAndTag`
Required: No

**failureReason**
The reason for the failure.
Type: String
Required: No

**imageId**
The image ID associated with the failure.
Type: ImageIdentifier (p. 58) object
Required: No

# ImageIdentifier

An object with identifying information for an Amazon ECR image.

## Contents

**imageDigest**

The `sha256` digest of the image manifest.

Type: String

Required: No

**imageTag**

The tag used for the image.

Type: String

Required: No

# Layer

An object representing an Amazon ECR image layer.

## Contents

**layerAvailability**
The availability status of the image layer. Valid values are `AVAILABLE` and `UNAVAILABLE`.
Type: String
Valid Values: `AVAILABLE | UNAVAILABLE`
Required: No

**layerDigest**
The `sha256` digest of the image layer.
Type: String
Pattern: `[a-zA-Z0-9-_+.]+:[a-fA-F0-9]+`
Required: No

**layerSize**
The size, in bytes, of the image layer.
Type: Long
Required: No

# LayerFailure

An object representing an Amazon ECR image layer failure.

## Contents

**failureCode**
The failure code associated with the failure.
Type: String
Valid Values: `InvalidLayerDigest | MissingLayerDigest`
Required: No

**failureReason**
The reason for the failure.
Type: String
Required: No

**layerDigest**
The layer digest associated with the failure.
Type: String
Length Constraints: Minimum length of 0. Maximum length of 1000.
Required: No

# ListImagesFilter

An object representing a filter on a ListImages (p. 41) operation.

## Contents

**tagStatus**

The tag status with which to filter your ListImages (p. 41) results. You can filter results based on whether they are `TAGGED` or `UNTAGGED`.

Type: String

Valid Values: `TAGGED` | `UNTAGGED`

Required: No

# Repository

An object representing a repository.

## Contents

**createdAt**
The date and time, in JavaScript date/time format, when the repository was created.
Type: Timestamp
Required: No

**registryId**
The AWS account ID associated with the registry that contains the repository.
Type: String
Pattern: `[0-9]{12}`
Required: No

**repositoryArn**
The Amazon Resource Name (ARN) that identifies the repository. The ARN contains the `arn:aws:ecr` namespace, followed by the region of the repository, the AWS account ID of the repository owner, the repository namespace, and then the repository name. For example, `arn:aws:ecr:region:012345678910:repository/test`.
Type: String
Required: No

**repositoryName**
The name of the repository.
Type: String
Length Constraints: Minimum length of 2. Maximum length of 256.
Pattern: `(?:[a-z0-9]+(?:[._-][a-z0-9]+)*/)*[a-z0-9]+(?:[._-][a-z0-9]+)*`
Required: No

**repositoryUri**
The URI for the repository. You can use this URI for Docker `push` and `pull` operations.
Type: String
Required: No

# Common Parameters

The following table lists the parameters that all actions use for signing Signature Version 4 requests. Any action-specific parameters are listed in the topic for that action. To view sample requests, see Examples of Signed Signature Version 4 Requests or Signature Version 4 Test Suite in the *Amazon Web Services General Reference*.

**Action**
The action to be performed.

Type: string

Required: Yes

**Version**
The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

**X-Amz-Algorithm**
The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: `AWS4-HMAC-SHA256`

Required: Conditional

**X-Amz-Credential**
The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key*/*YYYYMMDD*/*region*/*service*/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

**X-Amz-Date**

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: `20120325T120000Z`.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services General Reference*.

Type: string

Required: Conditional

**X-Amz-Security-Token**

The temporary security token that was obtained through a call to AWS Security Token Service. For a list of services that support AWS Security Token Service, go to Using Temporary Security Credentials to Access AWS in *Using Temporary Security Credentials*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

**X-Amz-Signature**

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

**X-Amz-SignedHeaders**

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

# Common Errors

This section lists the common errors that all actions return. Any action-specific errors are listed in the topic for the action.

**IncompleteSignature**
The request signature does not conform to AWS standards.

HTTP Status Code: 400

**InternalFailure**
The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

**InvalidAction**
The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

**InvalidClientTokenId**
The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

**InvalidParameterCombination**
Parameters that must not be used together were used together.

HTTP Status Code: 400

**InvalidParameterValue**
An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

**InvalidQueryParameter**
The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

**MalformedQueryString**
The query string contains a syntax error.

HTTP Status Code: 404

**MissingAction**
The request is missing an action or a required parameter.

HTTP Status Code: 400

**MissingAuthenticationToken**
The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

**MissingParameter**
A required parameter for the specified action is not supplied.

HTTP Status Code: 400

**OptInRequired**
The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

**RequestExpired**
The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

**ServiceUnavailable**
The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

**Throttling**
The request was denied due to request throttling.

HTTP Status Code: 400

**ValidationError**
The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400