

---

# IAM Roles Anywhere

## User Guide



## **IAM Roles Anywhere: User Guide**

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

What is IAM Roles Anywhere? .....	1
IAM Roles Anywhere concepts .....	1
Accessing IAM Roles Anywhere .....	1
Creating a trust anchor and profile .....	3
Step 1: Establish trust .....	3
Step 2: Configure roles .....	4
Next steps .....	5
IAM Roles Anywhere Authentication .....	6
Signing process .....	7
Task 1: Create a canonical request .....	7
Task 2: Create a string to sign .....	9
Task 3: Calculate the signature .....	10
Task 4: Add the signature to the HTTP request .....	10
CreateSession .....	10
Request Syntax .....	11
URI Request Parameters .....	11
Request Body .....	11
Response Syntax .....	12
Response Elements .....	12
Credentials Object .....	13
The relationship between CreateSession and AssumeRole .....	13
Obtaining temporary security credentials .....	15
Synopsis .....	15
Options .....	16
Output .....	16
Examples .....	17
Credential Helper on GitHub .....	17
Credential Helper Changelog .....	17
CredentialHelper version 1.0.4 .....	17
CredentialHelper version 1.0.3 .....	17
CredentialHelper version 1.0.2 .....	18
CredentialHelper version 1.0.1 .....	18
IAM Roles Anywhere trust model .....	19
Role trusts .....	19
Signature validation .....	20
Trust policy .....	20
Source Identity Rules .....	24
Revocation .....	24
Public key infrastructure .....	25
Certificate authorities .....	25
Certificates .....	25
Key management .....	25
Security .....	26
Workload identities .....	26
Data protection .....	27
Encryption in transit .....	27
Key management .....	25
Inter-network traffic privacy .....	28
Identity and access management .....	28
Audience .....	28
Authenticating with identities .....	29
Managing access using policies .....	31
How AWS Identity and Access Management Roles Anywhere works with IAM .....	32
Identity-based policy examples .....	37

---

Troubleshooting .....	39
Using service-linked roles .....	41
Resilience .....	43
Monitoring .....	45
Monitoring with CloudWatch .....	45
Monitoring events .....	46
Trust anchor certificate expiration event .....	47
Intermediate or end-entity certificate expiration event .....	47
Responding to an event .....	47
CloudTrail logs .....	48
IAM Roles Anywhere information in CloudTrail .....	48
Understanding IAM Roles Anywhere log file entries .....	49
Monitoring with IAM Roles Anywhere subjects .....	50
AWS CloudFormation resources .....	52
IAM Roles Anywhere and AWS CloudFormation templates .....	52
Learn more about AWS CloudFormation .....	52
VPC endpoints (AWS PrivateLink) .....	53
Considerations for IAM Roles Anywhere VPC endpoints .....	53
Creating an interface VPC endpoint for IAM Roles Anywhere .....	53
Creating a VPC endpoint policy for IAM Roles Anywhere .....	54
Quotas .....	55
Throttling .....	56

# What is AWS Identity and Access Management Roles Anywhere?

You can use AWS Identity and Access Management Roles Anywhere to obtain [temporary security credentials in IAM](#) for workloads such as servers, containers, and applications that run outside of AWS. Your workloads can use the same [IAM policies](#) and [IAM roles](#) that you use with AWS applications to access AWS resources. Using IAM Roles Anywhere means you don't need to manage long-term credentials for workloads running outside of AWS.

To use IAM Roles Anywhere, your workloads must use X.509 certificates issued by your [certificate authority \(CA\)](#). You register the CA with IAM Roles Anywhere as a trust anchor to establish trust between your public-key infrastructure (PKI) and IAM Roles Anywhere. You can also use AWS Private Certificate Authority (AWS Private CA) to create a CA and then use that to establish trust with IAM Roles Anywhere. AWS Private CA is a managed private CA service for managing your CA infrastructure and your private certificates. For more information, see [What is AWS Private CA](#).

## Topics

- [IAM Roles Anywhere concepts \(p. 1\)](#)
- [Accessing IAM Roles Anywhere \(p. 1\)](#)

## IAM Roles Anywhere concepts

Learn the basic terms and concepts used in IAM Roles Anywhere.

### • Trust anchors

You establish trust between IAM Roles Anywhere and your certificate authority (CA) by creating a *trust anchor*. A trust anchor is either a reference to [AWS Private CA](#) or another CA certificate. Your workloads outside of AWS authenticate with the trust anchor using certificates issued by the trusted CA in exchange for temporary AWS credentials. For more information, see [IAM Roles Anywhere trust model \(p. 19\)](#).

### • Roles

An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. A role is intended to be assumable by anyone who needs it. For IAM Roles Anywhere to be able to assume a role and deliver temporary AWS credentials, the role must trust the IAM Roles Anywhere service principal. For more information, see [the section called "Role trusts" \(p. 19\)](#).

### • Profiles

To specify which roles IAM Roles Anywhere assumes and what your workloads can do with the temporary credentials, you create a profile. In a profile, you can define permissions with IAM managed policies to limit the permissions for a created session.

## Accessing IAM Roles Anywhere

### AWS Management Console

You can manage your IAM Roles Anywhere resources using the browser-based console at <https://console.aws.amazon.com/rolesanywhere/>.

### **AWS Command Line Tools**

You can use the AWS command line tools to issue commands at your system command line to perform IAM Roles Anywhere and other AWS tasks. This can be faster and more convenient than using the console. The command line tools can be useful if you want to build scripts to perform AWS tasks.

AWS provides the [AWS Command Line Interface \(AWS CLI\)](#). For information about installing and using the AWS CLI, see the [AWS Command Line Interface User Guide](#) .

### **AWS SDKs**

The AWS software development kits (SDKs) consist of libraries and sample code for various programming languages and platforms including Java, Python, Ruby, .NET, iOS and Android, and others. The SDKs include tasks such as cryptographically signing requests, managing errors, and retrying requests automatically. For more information about the AWS SDKs, including how to download and install them, see [Tools for Amazon Web Services](#).

# Creating a trust anchor and profile in AWS Identity and Access Management Roles Anywhere

To use AWS Identity and Access Management Roles Anywhere for authentication to AWS from your workloads that run outside of AWS such as servers, containers, and applications, you first create a trust anchor and profile through the IAM Roles Anywhere console.

You establish trust between IAM Roles Anywhere and your certificate authority (CA) by creating a *trust anchor*. A trust anchor is either a reference to [AWS Private Certificate Authority \(AWS Private CA\)](#) or another CA certificate. You can create trust anchors for each certificate authority you want to trust.

To specify which roles IAM Roles Anywhere assumes and what your workloads can do with the temporary credentials, you create a profile. In a profile, you can define permissions with IAM managed policies.

## Topics

- [Step 1: Establish trust \(p. 3\)](#)
- [Step 2: Configure roles \(p. 4\)](#)
- [Next steps \(p. 5\)](#)

## Step 1: Establish trust

The first step of using IAM Roles Anywhere is creating a trust anchor, which requires you to reference a certificate authority (CA) that IAM Roles Anywhere will use to validate your authentication requests. Both root and intermediate CAs can be used as trust anchors. You can use either a AWS Private CA resource in your account or upload your own CA certificate. Note that CA certificates that are used as trust anchors have to satisfy certain constraints. For more information, see [Signature verification](#).

### To set up a certificate authority (CA)

- Do one of the following:
  - To use a AWS Private CA resource, open the [AWS Private CA console](#). Follow the instructions in the [AWS Private CA User Guide](#).
  - To use another CA, follow the instructions provided by the CA. You provide the certificate body in a later step.

### To create a trust anchor

1. Sign in to the [IAM Roles Anywhere console](#).
2. Choose **Create a trust anchor**.
3. In **Trust anchor name**, enter a name for the trust anchor.
4. For **Certificate authority (CA) source**, do one of the following:
  - To use an AWS Private CA resource, choose **AWS Private CA**. In the **AWS Private CA** table, choose the AWS Private CA resource.
  - To use another CA, choose **External certificate bundle**. In **External certificate bundle**, paste your CA certificate body. The certificate must be in Privacy Enhanced Mail (PEM) format.

5. (Optional) Add metadata to the trust anchor by attaching tags as key-value pairs. For more information, see [Tagging AWS resources](#).
6. Choose **Create a trust anchor**.

## Step 2: Configure roles

Before you can create an IAM Roles Anywhere profile, you need at least one IAM role that trusts the IAM Roles Anywhere service principal. Then you can create a profile that lists the roles IAM Roles Anywhere assumes. In a profile, you can also limit the permissions for a created session with IAM managed policies.

### To configure a role to trust IAM Roles Anywhere

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. On the IAM roles page, choose the role you want to use.
3. On the **Trust relationships** tab, choose **Edit trust policy**.
4. Update the trust policy to include `rolesanywhere.amazonaws.com` as shown below.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rolesanywhere.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    }
  ]
}
```

#### Important

Without a Condition statement present in a role trust policy, any valid certificate from the CA used as the trust anchor, or CAs subordinate to that trust anchor may be used to assume a role via IAM roles anywhere. We recommend you use Condition statements on both the subject and issuer attributes to ensure that only certificates that you intend to be able to assume a role can do so.

For information about editing role trust policies, see [Modifying a role \(console\)](#) in the *IAM User Guide*.

### To create a profile

1. Sign in to the [IAM Roles Anywhere console](#).



2. Choose **Create a profile**.
3. In **Profile name**, enter a name for the profile.
4. Under **Role**, choose the role you updated the trust policy for.
5. (Optional) Configure session policies by choosing up to 10 managed policies or write an inline policy.

Session policies limit the permissions for a created session, but do not grant permissions. For more information, see [Session policies](#).

6. (Optional) Add metadata to the profile by attaching tags as key–value pairs. For more information, see [Tagging AWS resources](#).
7. Choose **Create a profile**.

## Next steps

You can now authenticate with IAM Roles Anywhere. Follow the instructions in [Obtaining temporary security credentials \(p. 15\)](#). Also consider [Monitoring with IAM Roles Anywhere subjects \(p. 50\)](#).

# Authentication mechanism in AWS Identity and Access Management Roles Anywhere

To provide credentials, AWS Identity and Access Management Roles Anywhere uses the [CreateSession API \(p. 10\)](#). The API authenticates requests with a signature using keys associated with the X.509 certificate, which was used for authentication. The API method of IAM Roles Anywhere, [CreateSession \(p. 10\)](#) acts like AssumeRole - exchanging the signature for a standard SigV4-compatible session credential.

To successfully authenticate, the following constraints must be satisfied:

- The signature attached to the request **MUST** be validated against the signing certificate (also attached to the request).
- The signing certificate **MUST** have a valid trust chain to a Certificate Authority (CA) certificate configured in the customer account.
- The target role for which credentials are issued **MUST** have an AssumeRolePolicyDocument that allows IAM Roles Anywhere service principal, `rolesanywhere.amazonaws.com`, to call `sts:AssumeRole`, `sts:TagSession`, and `sts:SetSourceIdentity`. For more information, see [Granting permissions to pass a role to a service](#) in the *IAM User Guide*.
- The target role for which credentials are issued **MAY** have additional Condition predicates in the AssumeRolePolicyDocument that restrict authorization based on attributes extracted from the X.509 Certificate (for example, Subject or Issuer).

The signature uses the same canonicalization mechanism as [Signature V4 \(SigV4\)](#), with the following changes and additions:

- The private key used to sign the request **MUST** be bound to an X.509 Certificate.
- The signing certificate **MUST** be a v3 certificate.
- The signing certificate **MUST** be attached to the request via the header `X-Amz-X509`, as Base64-encoded Distinguished Encoding Rules (DER) data.
- The relevant headers - `X-Amz-X509` and `X-Amz-X509-Chain` (if applicable) **MUST** be included in the signed headers field of the `Authorization` header.
- The `X-Amz-X509-Chain` header **MUST** be encoded as comma-delimited, base64-encoded DER.
- The `X-Amz-X509-Chain` header **MUST NOT** exceed the maximum depth of 5 certificates.
- The signing certificate's serial number **MUST** be included in the Credential portion of the Scope field of the `Authorization` header.

RSA and EC keys are supported; RSA keys are used with the RSA PKCS# v1.5 signing algorithm. EC keys are used with the ECDSA.

## Topics

- [Signing process \(p. 7\)](#)

- [CreateSession API \(p. 10\)](#)

## Signing process

The signing process is identical to SigV4, with the exception of the keys used, the signature algorithm, and the addition of headers related to the X.509 certificate and trust chain. For more information, see [Signature Version 4 signing process](#), which should be treated as authoritative unless specifically addressed in this user guide.

### Topics

- [Task 1: Create a canonical request \(p. 7\)](#)
- [Task 2: Create a string to sign \(p. 9\)](#)
- [Task 3: Calculate the signature \(p. 10\)](#)
- [Task 4: Add the signature to the HTTP request \(p. 10\)](#)

## Task 1: Create a canonical request

To begin the signing process, create a string that includes information from your request in a standardized (canonical) format. This ensures that when AWS receives the request, it can calculate the same signature that you calculated.

Follow the steps below to create a canonical version of the request. Otherwise, your version and the version calculated by AWS won't match, and the request will be denied.

The following example shows the pseudocode to create a canonical request.

### Example Canonical request pseudocode

```
CanonicalRequest =  
  HttpRequestMethod + '\n' +  
  CanonicalUri + '\n' +  
  CanonicalQueryString + '\n' +  
  CanonicalHeaders + '\n' +  
  SignedHeaders + '\n' +  
  Lowercase(Hex(SHA256(RequestPayload)))
```

A canonical request has the above structure, in which specific elements of the request are transformed into a canonical value and concatenated together with other elements, joined with a newline character.

The following examples shows how to construct the canonical form of a request to IAM Roles Anywhere. The original request might look like this as it is sent from the client to AWS.

### Example Request

```
POST /sessions HTTP/1.1  
Host: rolesanywhere.us-east-1.amazonaws.com  
Content-Type: application/json  
X-Amz-Date: 20211103T120000Z  
X-Amz-X509: {base64-encoded DER data}  
{  
  "durationSeconds": number,  
  "profileArn": string,
```

```
"roleArn": string,  
"sessionName": string,  
"trustAnchorArn": string,  
}
```

To create a canonical request, concatenate the following components from each step into a single string:

1. The `HttpRequestMethod` is the verb of the HTTP request, in upper case. From the example, POST.
2. The `CanonicalUri` is the path of the request up until the query string delimiter `?`. From the example, `/sessions`. The path MUST be normalized according to RFC 3986, with redundant and relative path components removed. Path segments MUST be URI-encoded twice.
3. The `CanonicalQueryString` is the string following `?` in the request URI. If the request does not include a query string, use an empty string (essentially, a blank line). The example request does not include a query string, therefore we will use an empty string. To construct the canonical query string, complete the following steps:
  - a. Sort the parameter names by character code point in ascending order. Parameters with duplicate names should be sorted by value. For example, a parameter name that begins with the uppercase letter F precedes a parameter name that begins with a lowercase letter b.
  - b. URI-encode each parameter name and value according to the following rules:
    - i. Do not URI-encode any of the unreserved characters that RFC 3986 defines: A-Z, a-z, 0-9, hyphen ( - ), underscore ( \_ ), period ( . ), and tilde ( ~ ).
    - ii. Percent-encode all other characters with `%XY`, where X and Y are hexadecimal characters (0-9 and uppercase A-F). For example, the space character must be encoded as `%20` (not using `+`, as some encoding schemes do) and extended UTF-8 characters must be in the form `%XY%ZA%BC`.
    - iii. Double-encode any equals ( = ) characters in parameter values.
  - c. Build the canonical query string by starting with the first parameter name in the sorted list.
  - d. For each parameter, append the URI-encoded parameter name, followed by the equals sign character ( = ), followed by the URI-encoded parameter value. Use an empty string for parameters that have no value.
  - e. Append the ampersand character ( & ) after each parameter value, except for the last value in the list.
4. The `CanonicalHeaders` is a string capturing the header key and value that are included in the signature. The field is structured as follows:

```
CanonicalHeaders =  
  CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... + CanonicalHeadersEntryN  
CanonicalHeadersEntry =  
  Lowercase(HeaderName) + ':' + TrimAll(HeaderValue) + '\n'
```

`Lowercase` represents a function that converts all characters to lowercase. The `TrimAll` function removes excess white space before and after values, and converts sequential spaces to a single space.

### Important

The signing certificate MUST be presented in the header `X-Amz-X509`, as base64-encoded Distinguished Encoding Rules (DER), and the `X-Amz-X509` header MUST be included in `CanonicalHeaders` and `SignedHeaders`

If the client is providing the chain of intermediate certificates, the `X-Amz-X509-Chain` MUST be added to the request as well.

Build the canonical headers list by sorting the (lowercase) headers by character code and then iterating through the header names. Construct each header according to the following rules:

- Append the lowercase header name followed by a colon.
  - Append a comma-separated list of values for that header. Do not sort the values in headers that have multiple values.
  - Append a new line ('\n').
5. The SignedHeaders is a list of the header names, sorted by lowercase character code, delimited by semi-colon. For example - host;content-type;x-amz-date;x-amz-x509
  6. A hash of the request payload is appended to the canonical request. The bytes of the request are encoded as UTF-8, hashed with SHA-256, the resulting bytes hex encoded, and finally lowercased.
  7. To construct the finished canonical request, combine all the components from each step as a single string. As noted, each component ends with a newline character. If you follow the canonical request pseudocode explained earlier, the resulting canonical request is shown in the following example.

#### Example Canonical request

```
POST
/sessions
content-type:application/json
host:rolesanywhere.us-east-1.amazonaws.com
x-amz-date:20211103T120000Z
x-amz-x509:{base64-encoded DER data}

content-type;host;x-amz-date;x-amz-x509
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

8. Create a digest (hash) of the canonical request with the same algorithm that you used to hash the payload. The bytes of the request are encoded as UTF-8, hashed with SHA-256, the resulting bytes hex encoded, and finally lowercased.

## Task 2: Create a string to sign

The “string to sign” is the actual input to the signing algorithm, and includes meta information about the request, along with the canonical request created in the previous step.

#### Example StringToSign pseudocode

```
StringToSign =
  Algorithm + '\n' +
  RequestDateTime + '\n' +
  CredentialScope + '\n' +
  HashedCanonicalRequest
```

#### The structure is as follows:

1. The Algorithm is a string that indicates how the signature is calculated. It is of the form AWS4-X509-[AlgorithmId]-[HASH], where AlgorithmId is either RSA or ECDSA, and MUST be supported by the key type associated with the signing certificate. For example, if the signing certificate has an RSA key, the full algorithm string will be AWS4-X509-RSA-SHA256.
2. The RequestDateTime is a string derived at time of the signing operation, at second granularity, in UTC, formatted as ISO8601 basic, YYYYMMDD'T'HHMMSS'Z'. For example, 20211101T121030Z.

3. The `CredentialScope` is structured field of the form `Date + '/' + Region + '/' + Service + '/aws4_request'`. The Region and service name strings must be UTF-8 encoded. For example, `20211101/us-east-1/rolesanywhere/aws4_request`.
4. Finally, append a newline followed by the `HashedCanonicalRequest` computed in the previous step.

## Task 3: Calculate the signature

```
Signature = HexEncode(SigningAlgorithm(CertPrivateKey, StringToSign))
```

## Task 4: Add the signature to the HTTP request

The signature derived from the previous step is added to the HTTP request in the `Authorization` header field. The `Authorization` header is attached to the request and validated for authentication. It is partitioned into multiple fields - signing algorithm, credentials, signed headers, and the actual signature. The header of the authentication mechanism based on X.509 differs from a SigV4 header in two ways:

- **Algorithm.** As described above, instead of `AWS4-HMAC-SHA256`, the algorithm field will have the values of the form `AWS4-X509-RSA-SHA256` or `AWS4-X509-ECDSA-SHA256`, depending on whether an RSA or Elliptic Curve algorithm is used. This, in turn, is determined by the key bound to the signing certificate.
- **Scope field/credentials.** As specified above, the serial number of the certificate used to sign the request will be in place of the Access Key ID (credential) in the `Scope` field.

**The structure of the field is as follows:**

```
Authorization: {Algorithm} Credential={CredentialString},  
SignedHeaders={SignedHeaders}, Signature={Signature}
```

1. The `Algorithm` is of the form `AWS4-X509-{RSA|ECDSA}-SHA256`. Examples - `AWS4-X509-RSA-SHA256`.
2. The `Credential` is constructed via `{SerialNumber}/{Scope}` where serial number is the decimal representation of the serial number of the signing certificate, and `Scope` is the value constructed as input to the `StringToSign`. For example:

```
Credential=1111122223333344444/20201105/us-east-1/rolesanywhere/aws4_request
```

3. The `SignedHeaders` is a comma-delimited list of the headers signed as part of the request.
4. The `Signature` is the hex encoded output of the previous step.

## CreateSession API

Returns a temporary security credentials for workloads who have been authenticated with IAM Roles Anywhere to access AWS resources.

## Request Syntax

```
POST /sessions HTTP/1.1
Content-type: application/json
{
  "durationSeconds": number,
  "profileArn": string,
  "roleArn": string,
  "trustAnchorArn": string,
}
```

## URI Request Parameters

The request does not use any URI parameters.

## Request Body

The request accepts the following data in JSON format.

### durationSeconds

The duration, in seconds, of the role session. The value specified can range from 900 seconds (15 minutes) up to 3600 seconds (1 hour).

Type: Number

Required: No

### profileArn

The Amazon Resource Name (ARN) of the profile.

Type: String

Required: Yes

### roleArn

The Amazon Resource Name (ARN) of the role to assume.

Type: String

Required: Yes

### sessionName

Reserved for future use.

Type: String

Required: No

### trustAnchorArn

The Amazon Resource Name (ARN) of the trust anchor.

Type: String

Required: Yes

## Response Syntax

```
HTTP/1.1 201
Content-type: application/json
{
  "credentialSet":[
    {
      "assumedRoleUser": {
        "arn": ARN,
        "assumedRoleId": String
      },
      "credentials":{
        "accessKeyId": String,
        "expiration": Timestamp,
        "secretAccessKey": String,
        "sessionToken": String
      },
      "packedPolicySize": Number,
      "roleArn": ARN,
      "sourceIdentity": String
    }
  ],
  "subjectArn": ARN
}
```

## Response Elements

If the action is successful, the service sends back an HTTP 201 response.

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

### assumedRoleUser

The Amazon Resource Name (ARN) and the assumed role ID, which are identifiers that you can use to refer to the resulting temporary security credentials.

Type: [AssumedRoleUser](#) object

### credentials

The temporary security credentials, which include an access key ID, a secret access key, and a security (or session) token.

Type: [Credentials Object \(p. 13\)](#)

### packedPolicySize

A percentage value that indicates the packed size of the session policies and session tags combined passed in the request. The request fails if the packed size is greater than 100 percent, which means the policies and tags exceeded the allowed space.

Type: Integer

Valid Range: Minimum value of 0

### sourceIdentity

The source identity is specified by the principal that is calling the CreateSession operation.

Type: String



Pattern: `[\p{L}\p{Z}\p{N}_ : / = + \ - @ ] +`  
`roleArn`

The Amazon Resource Name (ARN) of the assumed role.

Type: String

`subjectArn`

The Amazon Resource Name (ARN) of the Subject resource.

The Subject resource records the history of the principal that is calling the `CreateSession` operation, including its first recorded authentication time, and last recorded authentication time.

Type: String

## Credentials Object

AWS credentials for API authentication.

`AccessKeyId`

The access key ID that identifies the temporary security credentials.

Type: String

Length Constraints: Minimum length of 16. Maximum length of 128.

Required: Yes

`Expiration`

The date on which the current credentials expire.

Type: Timestamp

Required: Yes

`SecretAccessKey`

The secret access key that can be used to sign requests.

Type: String

Required: Yes

`SessionToken`

The token that users must pass to the service API to use the temporary credentials.

Type: String

Required: Yes

## The relationship between `CreateSession` and `AssumeRole`

`CreateSession` is an x509 wrapper around [AssumeRole](#). The temporary session credentials are delivered to RolesAnywhere by `AssumeRole`, and then passed on without modification in the result of

CreateSession. CreateSession is not included in any SDK or client as there is not yet native SDK or client support for CreateSession's signing process.

# Obtaining temporary security credentials from AWS Identity and Access Management Roles Anywhere

To obtain temporary security credentials from AWS Identity and Access Management Roles Anywhere, use the credential helper tool that IAM Roles Anywhere provides. This tool is compatible with the `credential_process` feature available across the language SDKs. The helper manages the process of creating a signature with the certificate and calling the endpoint to obtain session credentials; it returns the credentials to the calling process in a standard JSON format. See [Temporary security credentials in IAM](#) for more information on session credentials.

To download the credential helper tool, use the following links.

Platform	Download URL	SHA256 checksum
Linux	<a href="https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Linux/aws_signing_helper">https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Linux/aws_signing_helper</a>	bc625c319d96f71c05d899eab04402dc63
Windows	<a href="https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Windows/aws_signing_helper.exe">https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Windows/aws_signing_helper.exe</a>	b5363e902b938a5fca5e627d1be51df656
Darwin	<a href="https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Darwin/aws_signing_helper">https://rolesanywhere.amazonaws.com/releases/1.0.4/X86_64/Darwin/aws_signing_helper</a>	41ea27f8db1603e93d092f6249cacc1256

Before you can obtain temporary credentials, you must have both an end-entity certificate from the certificate authority and a trust anchor in IAM Roles Anywhere. For more information, see [Creating a trust anchor and profile](#) (p. 3).

## Synopsis

```
./aws_signing_helper credential-process \  
  --certificate  
  [--endpoint]  
  [--region]  
  [--intermediates]  
  --private-key  
  --profile-arn  
  --role-arn  
  [--session-duration]  
  --trust-anchor-arn  
  [--with-proxy]  
  [--no-verify-ssl]
```

## Options

`--certificate` (string)

Path to certificate file

`--endpoint` (string)

The IAM Roles Anywhere endpoint for the region. For a list of endpoints, see [Service endpoints and quotas](#).

`--region` (string)

Signing region.

`--intermediates` (string)

Path to intermediate certificate bundle.

`--private-key` (string)

Path to private key file.

`--profile-arn` (string)

Profile to pull policies from.

`--role-arn` (string)

Target role to assume.

`--session-duration` (int)

Duration, in seconds, for the resulting session (default 3600).

`--trust-anchor-arn` (string)

Trust anchor to use for authentication.

`--with-proxy`

To use the tool with a proxy. This is a boolean flag. Note that you will have to set the `HTTPS_PROXY` environment variable with the address of the proxy server.

`--no-verify-ssl`

To disable SSL verification. This is a boolean flag.

### Important

Note that this disables TLS host authentication, and can open the connection to man-in-the-middle attacks. This option should only be used under specific, tightly controlled scenarios, such as debugging proxy connections.

## Output

The credential helper tool will return a JSON containing the credentials. This format allows the credentials to be consumed by the [external credential process](#) supported by the `credential_process`.

```
{
  "Version":1,
  "AccessKeyId": String,
  "SecretAccessKey": String,
  "SessionToken": String,
```

```
} "Expiration": Timestamp
```

## Examples

### Example Obtain temporary security credentials

To obtain temporary security credentials, use the following command.

```
$ ./aws_signing_helper credential-process \  
  --certificate /path/to/certificate \  
  --private-key /path/to/private-key \  
  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID \  
  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \  
  --role-arn arn:aws:iam::account:role/role-name-with-path
```

### Example Use temporary security credentials with AWS SDKs and the AWS CLI

To use temporary security credentials with AWS SDKs and the AWS CLI, you can configure the credential helper tool as a credential process. For more information, see [Sourcing credentials with an external process](#).

The following example shows a the config file that sets the helper tool as the credential process.

```
[profile developer]  
  credential_process = ./aws_signing_helper credential-process --certificate /  
path/to/certificate --private-key /path/to/private-key --trust-anchor-  
arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID --profile-  
arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID --role-arn  
arn:aws:iam::account:role/role-name-with-path
```

## Credential Helper on GitHub

The source code for the credential helper is available on [GitHub](#) so that you can adapt the helper to your needs. We encourage you to submit pull requests for changes that you would like to have included. However, AWS doesn't provide support for running modified copies of this software.

## Credential Helper Changelog

### CredentialHelper version 1.0.4

On January 17, 2023, AWS IAM Roles Anywhere released Credential Helper version 1.0.4. As a part of this release, some bugs specific to the serve command were fixed.

### CredentialHelper version 1.0.3

On December 5, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.3. As a part of this release, the tool now supports the update and serve commands.

## CredentialHelper version 1.0.2

On September 8, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.2. As a part of this release, the tool now sets the minimum TLS version to 1.2.

## CredentialHelper version 1.0.1

On July 14, 2022, AWS IAM Roles Anywhere released Credential Helper version 1.0.1. As a part of this release, the tool now has better error handling.

# Trust model in AWS Identity and Access Management Roles Anywhere

AWS Identity and Access Management Roles Anywhere works by bridging the trust model of IAM and Public Key Infrastructure (PKI). The model connects the role, the IAM Roles Anywhere service principal, and identities encoded in X509 certificates, that are issued by a Certificate Authority (CA).

## Role trusts

To use an IAM role with IAM Roles Anywhere, you must create a trust relationship with the IAM Roles Anywhere service principal `rolesanywhere.amazonaws.com`. To create the trust relationship, you create a *trust policy*, a JSON policy document. The policy must grant the permissions:

- `sts:AssumeRole`
- `sts:SetSourceIdentity`
- `sts:TagSession`

Sessions issued by IAM Roles Anywhere have the source identity set to the common name(s) (CN) of the subject(s) you use in end-entity certificate(s) authenticating to the target role. IAM Roles Anywhere extracts values from the subject, issuer, and Subject Alternative Name (SAN) fields of the authenticating certificate and makes them available for policy evaluation via the `sourceIdentity` and `principal` tags. The `sts:SourceIdentity` key is present in the request when IAM Roles Anywhere initially sets the source identity while assuming a role. You can apply more authorization constraints by using condition clauses in the policy statement. The tags will also be evaluated against policies on any resources accessed with the issued session.

To examine the contents of a certificate, use the following command:

```
$openssl x509 -text -noout -in certificate.pem
```

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    1f:71:c5:11:4a:11:9f:c0:cc:5a:5a:52:fb:37:20:ad
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=Amazon, OU=IAM, ST=Washington, CN=Roles Anywhere, L=Seattle
  Validity
    Not Before: Apr 20 14:13:43 2022 GMT
    Not After : Jan 10 15:13:43 2023 GMT
  Subject: CN=awsUser
  # remainder omitted...
```

The value `awsUser` becomes the source identity. The values from the `Issuer` field become principal tags in the resulting session.

For more information, see:

- [IAM roles](#)

- [IAM roles terms and concepts: trust policy](#)
- [Policies and permissions in IAM](#)
- [Things to know about source identity](#)
- [Passing session tags in AWS Security Token Service](#)

## Signature validation

To authenticate a request for credentials, IAM Roles Anywhere validates the incoming signature by using the signature validation algorithm required by the key type of the certificate, for example RSA or ECDSA. After validating the signature, IAM Roles Anywhere checks that the certificate was issued by a certificate authority configured as a trust anchor in the account using algorithms defined by public key infrastructure X.509 (PKIX) standards.

End entity certificates must satisfy the following constraints to be used for authentication:

- The certificates **MUST** be X.509v3.
- Basic constraints **MUST** include `CA: false`.
- The key usage **MUST** include `Digital Signature`.
- The signing algorithm **MUST** include SHA256 or stronger. MD5 and SHA1 signing algorithms are rejected.

Certificates used as trust anchors must satisfy the same requirements for signature algorithm, but with the following differences:

- The key usage **MUST** include `Certificate Sign`, and **MAY** include `CRL Sign`. Certificate Revocation Lists (CRLs) are an optional feature of IAM Roles Anywhere.
- Basic constraints **MUST** include `CA: true`.

## Trust policy

Temporary credentials for IAM roles are issued to IAM Roles Anywhere clients via the API method `CreateSession`. In order for the call to be authorized, the target role of the `CreateSession` API call must have an Assume Role Policy Document to trust the IAM Roles Anywhere service principal.

### Important

It is also recommended to have additional condition statements to further restrict authorization based on attributes that are extracted from the X.509 certificate.

The `Subject`, `Issuer`, and `Subject Alternative Name (SAN)` fields from X509 certificates are extracted and used as `PrincipalTag` elements in the session.

### Certificate Subject fields mapping

IAM Roles Anywhere maps Relative Distinguished Names (RDNs) from an authenticating certificate's `Subject` into distinct `PrincipalTag` elements in the session.

The following example shows a trust policy that adds a condition based on the `Subject Common Name (CN)` of the certificate. For example, if the identity asserted in the certificate is `Alice`, a condition can be created on the `CN` of the `Subject`.

```
{
```



```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": "rolesanywhere.amazonaws.com",
    "Action": [
      "sts:AssumeRole",
      "sts:TagSession",
      "sts:SetSourceIdentity"
    ],
    "Condition": {
      "StringEquals": {
        "aws:PrincipalTag/x509Subject/CN": "Alice"
      }
    }
  }
]
```

### Certificate Issuer fields mapping

IAM Roles Anywhere maps RDNs from an authenticating certificate's Issuer into distinct `PrincipalTag` elements in the session.

The following example shows a trust policy that adds a condition based on the Issuer Common Name (CN) of the certificate. For example, if the issuer identity asserted in the certificate is Bob, a condition can be created on the CN of the Issuer.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "rolesanywhere.amazonaws.com",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/x509Issuer/CN": "Bob"
        }
      }
    }
  ]
}
```

### Certificate Subject Alternative Name (SAN) fields mapping

X509v3 certificates may include an extension to define additional identities, call Subject Alternative Names (SANs). The SAN can have multiple values, each of which can be one of nine types - Domain Name System (DNS) names, Uniform Resource Indicators (URIs), Directory Names, IP addresses, "Other" names, EDI party names, X400 addresses, RFC 822 names, or registered OIDs. IAM Roles Anywhere will map the **first** value of the following types: DNS Names, Directory Name (DN), and URI Names. The resulting tags will be prefixed with `x509SAN`, with a corresponding type code. The type code will be one of DNS, URI, or Name. For values of type DN, the individual RDNs will be parsed, as with Subject and Issuer.

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
```

```
1f:71:c5:11:4a:11:9f:c0:cc:5a:5a:52:fb:37:20:ad
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=US, O=Amazon, OU=IAM, ST=Washington, CN=RolesAnywhere, L=Seattle
Validity
  Not Before: Apr 20 14:13:43 2022 GMT
  Not After : Jan 10 15:13:43 2023 GMT
Subject: CN=Alice
X509v3 extensions:
  X509v3 Subject Alternative Name:
    DNS:example.com,
    URI:spiffe://example.com/workload/alice,
    DirName:/CN=Alice
# remainder omitted...
```

The complete set of session principal tags, including the SAN fields, will look like the following:

```
"aws:PrincipalTag/x509Subject/CN": "Alice"
"aws:PrincipalTag/x509Issuer/C": "US"
"aws:PrincipalTag/x509Issuer/O": "Amazon"
"aws:PrincipalTag/x509Issuer/OU": "IAM"
"aws:PrincipalTag/x509Issuer/ST": "Washington"
"aws:PrincipalTag/x509Issuer/L": "Seattle"
"aws:PrincipalTag/x509Issuer/CN": RolesAnywhere
"aws:PrincipalTag/x509SAN/DNS": "example.com"
"aws:PrincipalTag/x509SAN/URI": "spiffe://example.com/workload/alice"
"aws:PrincipalTag/x509SAN/Name/CN": "Alice"
```

The following example shows a trust policy that adds a condition based on the SAN fields of the certificate.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "rolesanywhere.amazonaws.com",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/x509SAN/DNS": "example.com",
          "aws:PrincipalTag/x509SAN/URI": "spiffe://example.com/workload/alice",
          "aws:PrincipalTag/x509SAN/Name/CN": "Alice"
        }
      }
    }
  ]
}
```

#### Note

[RFC 5280](#) allows certificates with empty subjects if and only if the SAN extension is present and marked critical. Certificates with empty subjects are NOT yet supported, since IAM Roles Anywhere uses the certificate subject as the key of the Subject resource to visualize and audit activities for certificates that are authenticated with IAM Roles Anywhere. For more information about Subject resource, see [Monitoring with IAM Roles Anywhere subjects \(p. 50\)](#).

**Using the `aws:SourceArn` or `aws:SourceAccount` or `sts:SourceIdentity` condition keys**

In general, it is strongly recommended that you use the [aws:SourceArn](#) or the [aws:SourceAccount](#) global condition keys or the [sts:SourceIdentity](#) condition key in your role trust policies. This combination of conditions implements least privilege permissions and prevents IAM Roles Anywhere from acting as a potential [confused deputy](#). When a client obtains temporary security credentials from IAM Roles Anywhere, the `aws:SourceArn` and `aws:SourceAccount` will be set based on the ARN of the trust anchor specified in the call to `CreateSession`.

To use the `aws:SourceArn` and `aws:SourceAccount` global condition keys, set the value to the Amazon Resource Name (ARN) or account of the trust anchor that you expect to use for authentication. As an example, consider the following role trust policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "rolesanywhere.amazonaws.com",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID"
          ]
        }
      }
    }
  ]
}
```

To use the `sts:SourceIdentity` condition key, set the source identity prefix and source identity value according to the following [rules \(p. 24\)](#). As an example, consider the following role trust policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "rolesanywhere.amazonaws.com",
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "sts:SourceIdentity": [
            "sourceIdentityPrefix:sourceIdentityValue"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": "rolesanywhere.amazonaws.com",
      "Action": [
        "sts:TagSession"
      ]
    }
  ]
}
```

```
}
```

## Source Identity Rules

We define a source identity prefix as follows:

- "CN=": the common name of the subject in the certificate is set and less than or equal to the length threshold.
- "ID=": the common name of the subject in the certificate is NOT set.
- "": (empty string) the common name of the subject in the certificate is set and greater than the length threshold but less than 64 characters.

The value following a source identity prefix is as follows:

- "CN=": the common name of the subject in the certificate.
- "ID=": the hex-encoded certificate serial number.
- "": the common name of the subject in the certificate.

The length threshold is 61 characters.

## Revocation

Certificate revocation is supported through the use of imported certificate revocation lists (CRLs). You can upload a CRL that is generated from your CA, and certificates used for authentication will be checked for their revocation status. Callbacks to CRL Distribution Points (CDPs) or Online Certificate Status Protocol (OCSP) endpoints are not supported.

# Public key infrastructure in AWS Identity and Access Management Roles Anywhere

AWS Identity and Access Management Roles Anywhere relies on public key infrastructure (PKI) to establish trust between your AWS account and certificate authorities that issue certificates to workloads running in your data centers. PKI involves the generation, distribution, and verification of digital certificates through public key encryption. Trust requires either uploading your CA's digital certificate as a trust anchor or referencing an existing [AWS Private Certificate Authority \(AWS Private CA\)](#). Once a trust anchor is created, you can use client certificates issued by that certificate to authenticate and receive temporary credentials from IAM Roles Anywhere.

## Certificate authorities

*Certificate authorities* are entities that are trusted to issue certificates. The CA issues signed digital certificates that affirm the identity of the certificate subject and bind that identity to the public key contained in the certificate. The CA signs a certificate by hashing the contents and then encrypting the hash with the private key related to the public key in the certificate. A client application such as a web browser that needs to affirm the identity of a subject uses the public key to verify the certificate signature. It then hashes the certificate contents and compares the hashed value to the decrypted signature to determine whether they match. Certificates can have constraints on the uses of the keys associated with the certificate. For more information about trust path validation, see [RFC 5280](#).

## Certificates

Certificates, specifically X.509 certificates, bind an identity to public key, using a signature generated from a corresponding private key.

## Key management

PKI uses a pair of keys to perform cryptographic operations such as encryption and generating digital signatures. One of the keys is public and is typically made available in an X.509 certificate. The other key is private and should be stored securely. Take care to ensure that your private keys are not shared.

# Security in AWS Identity and Access Management Roles Anywhere

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are 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 AWS Identity and Access Management Roles Anywhere, 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 AWS Identity and Access Management Roles Anywhere. The following topics show you how to configure IAM Roles Anywhere to meet your security and compliance objectives. You can also learn how to use other AWS services that help you to monitor and secure your IAM Roles Anywhere resources.

## Topics

- [Mapping identities to your workloads with AWS Identity and Access Management Roles Anywhere \(p. 26\)](#)
- [Data protection in AWS Identity and Access Management Roles Anywhere \(p. 27\)](#)
- [Identity and access management for AWS Identity and Access Management Roles Anywhere \(p. 28\)](#)
- [Resilience in AWS Identity and Access Management Roles Anywhere \(p. 43\)](#)

## Mapping identities to your workloads with AWS Identity and Access Management Roles Anywhere

A key element to using IAM Roles Anywhere is managing how identities are assigned to workloads. Certificates are issued to compute instances (servers, containers), in which the identity is encoded as the Subject of the certificate. The subject may be a simple Common Name (CN), a Fully Qualified Distinguished Name (FQDN), that contains information about organizational structure, or a simple hostname. Alternatively, a standard such as [SPIFFE](#) could be used to create a hierarchical namespace for the workload identities.

IAM Roles Anywhere lets the workload use the certificate to obtain temporary credentials instead of issuing Access Key IDs and Secret Access Keys, and the identity in the subject is encoded in the session credentials in a way that can be used in resource policies. For example, if a certificate has a Subject with

CN=Alice, the value is added to the session as a `PrincipalTag: aws:PrincipalTag/x509Subject/CN`.

## Data protection in AWS Identity and Access Management Roles Anywhere

The AWS [shared responsibility model](#) applies to data protection in AWS Identity and Access Management Roles Anywhere. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center (successor to AWS Single Sign-On) or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with IAM Roles Anywhere or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

### Encryption in transit

IAM Roles Anywhere provides secure and private endpoints for encrypting data in transit. The secure and private endpoints allow AWS to protect the integrity of API requests to IAM Roles Anywhere. AWS requires API calls to be signed by the caller using a secret access key. This requirement is stated in the [Signature Version 4 Signing Process \(Sigv4\)](#).

### Key management

Authenticating to IAM Roles Anywhere requires the use of asymmetric (public/private) key pairs. IAM Roles Anywhere does not hold customer private keys. Those remain on customer instances in data centers. Care should be taken to minimize the risk of accidental disclosure.

- Use OS file system permissions to restrict read access to the owning user.

- Never check keys into source control. Store them separately from source code to reduce the risk of accidentally including them in a change set. If possible, consider the use of a secure storage mechanism.

## Inter-network traffic privacy

When accessing AWS APIs and resources from your data center, be aware of how the traffic is routed. Connectivity may occur via Network Address Translation (NAT) at the data center outbound firewall, or it may occur through Virtual Private Network (VPN).

# Identity and access management for AWS Identity and Access Management Roles Anywhere

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 IAM Roles Anywhere resources. IAM is an AWS service that you can use with no additional charge.

### Topics

- [Audience \(p. 28\)](#)
- [Authenticating with identities \(p. 29\)](#)
- [Managing access using policies \(p. 31\)](#)
- [How AWS Identity and Access Management Roles Anywhere works with IAM \(p. 32\)](#)
- [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#)
- [Troubleshooting AWS Identity and Access Management Roles Anywhere identity and access \(p. 39\)](#)
- [Using service-linked roles for AWS Identity and Access Management Roles Anywhere \(p. 41\)](#)

## Audience

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

**Service user** – If you use the IAM Roles Anywhere service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more IAM Roles Anywhere 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 IAM Roles Anywhere, see [Troubleshooting AWS Identity and Access Management Roles Anywhere identity and access \(p. 39\)](#).

**Service administrator** – If you're in charge of IAM Roles Anywhere resources at your company, you probably have full access to IAM Roles Anywhere. It's your job to determine which IAM Roles Anywhere 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 IAM Roles Anywhere, see [How AWS Identity and Access Management Roles Anywhere works with IAM \(p. 32\)](#).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to IAM Roles Anywhere. To view example IAM Roles Anywhere identity-based policies that you can use in IAM, see [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#).



## 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*.

## Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center (successor to AWS Single Sign-On). You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User 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 permission 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 [Actions, Resources, and Condition Keys for AWS Identity and Access Management Roles Anywhere](#) 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 IAM 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.

## Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

## 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 AWS Identity and Access Management Roles Anywhere works with IAM

Before you use IAM to manage access to IAM Roles Anywhere, learn what IAM features are available to use with IAM Roles Anywhere.

### IAM features you can use with AWS Identity and Access Management Roles Anywhere

IAM feature	IAM Roles Anywhere support
<a href="#">Identity-based policies (p. 33)</a>	Yes
<a href="#">Resource-based policies (p. 33)</a>	No

IAM feature	IAM Roles Anywhere support
<a href="#">Policy actions (p. 34)</a>	Yes
<a href="#">Policy resources (p. 34)</a>	Yes
<a href="#">Policy condition keys (p. 35)</a>	Yes
<a href="#">ACLs (p. 35)</a>	No
<a href="#">ABAC (tags in policies) (p. 36)</a>	Partial
<a href="#">Temporary credentials (p. 36)</a>	Yes
<a href="#">Principal permissions (p. 36)</a>	Yes
<a href="#">Service roles (p. 37)</a>	Yes
<a href="#">Service-linked roles (p. 37)</a>	Yes

To get a high-level view of how IAM Roles Anywhere and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

## Identity-based policies for IAM Roles Anywhere

Supports identity-based policies	Yes
----------------------------------	-----

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*.

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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

### Identity-based policy examples for IAM Roles Anywhere

To view examples of IAM Roles Anywhere identity-based policies, see [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#).

## Resource-based policies within IAM Roles Anywhere

Supports resource-based policies	No
----------------------------------	----

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.

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, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They 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*.

## Policy actions for IAM Roles Anywhere

Supports policy actions	Yes
-------------------------	-----

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.

To see a list of IAM Roles Anywhere actions, see [Actions Defined by AWS Identity and Access Management Roles Anywhere](#) in the *Service Authorization Reference*.

Policy actions in IAM Roles Anywhere use the following prefix before the action:

```
rolesanywhere
```

To specify multiple actions in a single statement, separate them with commas.

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

To view examples of IAM Roles Anywhere identity-based policies, see [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#).

## Policy resources for IAM Roles Anywhere

Supports policy resources	Yes
---------------------------	-----

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": "*"

```

To see a list of IAM Roles Anywhere resource types and their ARNs, see [Resources Defined by AWS Identity and Access Management Roles Anywhere](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by AWS Identity and Access Management Roles Anywhere](#).

To view examples of IAM Roles Anywhere identity-based policies, see [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#).

## Policy condition keys for IAM Roles Anywhere

Supports service-specific policy condition keys	Yes
---	-----

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*.

To see a list of IAM Roles Anywhere condition keys, see [Condition Keys for AWS Identity and Access Management Roles Anywhere](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions Defined by AWS Identity and Access Management Roles Anywhere](#).

To view examples of IAM Roles Anywhere identity-based policies, see [Identity-based policy examples for AWS Identity and Access Management Roles Anywhere \(p. 37\)](#).

## Access control lists (ACLs) in IAM Roles Anywhere

Supports ACLs	No
---------------	----

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

## Attribute-based access control (ABAC) with IAM Roles Anywhere

Supports ABAC (tags in policies)	Partial
----------------------------------	---------

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see [What is ABAC?](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

## Using Temporary credentials with IAM Roles Anywhere

Supports temporary credentials	Yes
--------------------------------	-----

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switching to a role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

## Cross-service principal permissions for IAM Roles Anywhere

Supports principal permissions	Yes
--------------------------------	-----

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 [Actions, Resources, and Condition Keys for AWS Identity and Access Management Roles Anywhere](#) in the *Service Authorization Reference*.



## Service roles for IAM Roles Anywhere

Supports service roles	Yes
------------------------	-----

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*.

### Warning

Changing the permissions for a service role might break IAM Roles Anywhere functionality. Edit service roles only when IAM Roles Anywhere provides guidance to do so.

## Service-linked roles for IAM Roles Anywhere

Supports service-linked roles	Yes
-------------------------------	-----

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 IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see [AWS services that work with IAM](#). Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

## Identity-based policy examples for AWS Identity and Access Management Roles Anywhere

By default, users and roles don't have permission to create or modify IAM Roles Anywhere 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 IAM Roles Anywhere, including the format of the ARNs for each of the resource types, see in the *Service Authorization Reference*.

### Topics

- [Policy best practices \(p. 37\)](#)
- [Using the IAM Roles Anywhere console \(p. 38\)](#)
- [Allow users to view their own permissions \(p. 38\)](#)

## Policy best practices

Identity-based policies determine whether someone can create, access, or delete IAM Roles Anywhere 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 IAM Roles Anywhere console

To access the AWS Identity and Access Management Roles Anywhere console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the IAM Roles Anywhere 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.

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.

To ensure that users and roles can still use the IAM Roles Anywhere console, also attach the IAM Roles Anywhere ConsoleAccess or ReadOnly AWS managed policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

## 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": "*"
  }
]
}
```

## Troubleshooting AWS Identity and Access Management Roles Anywhere identity and access

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

### Topics

- [I am not authorized to perform an action in IAM Roles Anywhere \(p. 39\)](#)
- [I am not authorized to perform iam:PassRole \(p. 40\)](#)
- [I want to view my access keys \(p. 40\)](#)
- [I'm an administrator and want to allow others to access IAM Roles Anywhere \(p. 40\)](#)
- [I want to allow people outside of my AWS account to access my IAM Roles Anywhere resources \(p. 40\)](#)

### I am not authorized to perform an action in IAM Roles Anywhere

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

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 does not have the fictional rolesanywhere: *GetWidget* permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the *my-example-widget* resource using the `rolesanywhere:GetWidget` action.

## 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 IAM Roles Anywhere.

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 IAM Roles Anywhere. 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 want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

### Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing access keys](#) in the *IAM User Guide*.

## I'm an administrator and want to allow others to access IAM Roles Anywhere

To allow others to access IAM Roles Anywhere, 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 IAM Roles Anywhere.

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 IAM Roles Anywhere 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 IAM Roles Anywhere supports these features, see [How AWS Identity and Access Management Roles Anywhere works with IAM \(p. 32\)](#).
- 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*.

## Using service-linked roles for AWS Identity and Access Management Roles Anywhere

AWS Identity and Access Management Roles Anywhere uses IAM [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to IAM Roles Anywhere. Service-linked roles are predefined by IAM Roles Anywhere and include all the permissions that the service requires to ensure the private certificate authorities you use as trust anchors can be accessed as part of authenticating with IAM Roles Anywhere. They also ensure that you receive auditing information regarding IAM Roles Anywhere. Service-linked roles are different from the roles that you configure for the service and obtain temporary credentials for.

A service-linked role makes setting up IAM Roles Anywhere easier because you don't have to manually add the necessary permissions. IAM Roles Anywhere defines the permissions of its service-linked roles. Unless defined otherwise, only IAM Roles Anywhere can assume its service-linked roles. The defined permissions include the trust policy and the permissions policy. The permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your IAM Roles Anywhere resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS services that work with IAM](#) and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

## Service-linked role permissions for IAM Roles Anywhere

IAM Roles Anywhere uses the service-linked role named **AWSServiceRoleForRolesAnywhere** which allows IAM Roles Anywhere to publish CloudWatch metrics and check the configuration of AWS Private CA in your account.

The **AWSServiceRoleForRolesAnywhere** service-linked role trusts the following services to assume the role:

- `rolesanywhere.amazonaws.com`

The role permissions policy named **AWSRolesAnywhereServicePolicy** allows IAM Roles Anywhere to complete the following actions on the specified resources:

- Action: `cloudwatch:PutMetricData` on the `AWS/RolesAnywhere` and `AWS/Usage` namespaces
- Action: `acmpca:GetCertificateAuthorityCertificate` on `AWS Private CAs`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see [Service-linked role permissions](#) in the *IAM User Guide*.

## Creating a service-linked role for IAM Roles Anywhere

You don't need to manually create a service-linked role. When you create your first trust anchor in the AWS Management Console, the AWS CLI, or the AWS API, IAM Roles Anywhere creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. Note that credentials will still be issued, but metrics will not be reported.

You can also use the IAM console to create a service-linked role when you have trust anchors in your account but no service-linked role. In the AWS CLI or the AWS API, create a service-linked role with the `rolesanywhere.amazonaws.com` service name. For more information, see [Creating a service-linked role](#) in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

## Editing a service-linked role for IAM Roles Anywhere

IAM Roles Anywhere does not allow you to edit the `AWSServiceRoleForRolesAnywhere` service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see [Editing a service-linked role](#) in the *IAM User Guide*.

## Deleting a service-linked role for IAM Roles Anywhere

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

### Note

If IAM Roles Anywhere is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

### To delete IAM Roles Anywhere resources used by the `AWSServiceRoleForRolesAnywhere`

- Delete all trust anchors in your account in all Regions that contain them.

### To manually delete the service-linked role using IAM

- For information about using the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForRolesAnywhere` service-linked role, see [Deleting a service-linked role](#) in the *IAM User Guide*.

## Supported regions for IAM Roles Anywhere service-linked roles

IAM Roles Anywhere supports using service-linked roles in all of the regions where the service is available. For more information, see [AWS regions and endpoints](#).

Region name	Region identity	Support in IAM Roles Anywhere
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Asia Pacific (Hong Kong)	ap-east-1	Yes
Asia Pacific (Jakarta)	ap-southeast-3	Yes
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
Europe (Milan)	eu-south-1	Yes
Europe (Stockholm)	eu-north-1	Yes
Africa (Cape Town)	af-south-1	Yes
South America (São Paulo)	sa-east-1	Yes
Middle East (Bahrain)	me-south-1	Yes

## Resilience in AWS Identity and Access Management Roles Anywhere

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, IAM Roles Anywhere offers several features to help support your data resiliency and backup needs.



# Monitoring AWS Identity and Access Management Roles Anywhere

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Identity and Access Management Roles Anywhere and your other AWS solutions. AWS provides the following monitoring tools to watch IAM Roles Anywhere, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the [Amazon CloudWatch User Guide](#).
- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see [Amazon EventBridge User Guide](#).
- *Amazon EventBridge* is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, Software-as-a-Service (SaaS) applications, and AWS services and routes that data to targets such as Lambda. This enables you to monitor events that happen in services, and build event-driven architectures. For more information, see the [Amazon EventBridge User Guide](#).
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the [AWS CloudTrail User Guide](#).

## Monitoring AWS Identity and Access Management Roles Anywhere with Amazon CloudWatch

You can monitor AWS Identity and Access Management Roles Anywhere using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](#).

For IAM Roles Anywhere, you might want to watch for `certificate` expiration and renew your certificates when your certificates are nearing expiration.

The IAM Roles Anywhere service reports the following metrics in the `AWS/RolesAnywhere` namespace.

Metric	Description
Success	Gets published every time <code>CreateSession</code> succeeds in returning credentials to the user.

Metric	Description
	Valid Dimensions: Operation, TrustAnchorArn Valid Statistic: Sum Units: Count
Failure	Gets published every time CreateSession fails to return credentials to the user. Valid Dimensions: Operation, ErrorType Valid Statistic: Sum Units: Count
DaysToExpiry	Gets published every time trust anchor certificates are within 45 days of expiry. This metric will be published at most once a day. Valid Dimensions: TrustAnchorArn Units: Integer

The following dimensions are supported for the IAM Roles Anywhere metrics.

Dimension	Description
Operation	The operation for which the metric applies to. This can only take on the value, CreateSession.
TrustAnchorArn	The ARN of the trust anchor that is relevant for this metric.
ErrorType	The type of error that CreateSession errors out with.

## Monitoring AWS Identity and Access Management Roles Anywhere events in Amazon EventBridge

You can monitor IAM Roles Anywhere events in [Amazon EventBridge](#). Events from IAM Roles Anywhere are delivered to EventBridge in near-real time. You can write simple rules to indicate which events are of interest to you and the automated actions to take when an event matches a rule. With EventBridge, you can use events to trigger targets including AWS Lambda functions, AWS Batch jobs, Amazon SNS topics, and many others. For more information, see [Creating Amazon EventBridge rules that react to events](#).

The following examples show events for IAM Roles Anywhere.

### Topics

- [Trust anchor certificate expiration event \(p. 47\)](#)
- [Intermediate or end-entity certificate expiration event \(p. 47\)](#)
- [Responding to an event \(p. 47\)](#)

## Trust anchor certificate expiration event

IAM Roles Anywhere sends daily expiration event for each trust anchor certificate that is within 45 days prior to expiration. You can use expiration events to configure Amazon SNS to send a text notification whenever IAM Roles Anywhere generates this event.

Expiration events have the following structure.

```
{
  "version": "0",
  "id": "9c95e8e4-96a4-ef3f-b739-b6aa5b193afb",
  "detail-type": "Roles Anywhere Trust Anchor Certificate Approaching Expiration Notification",
  "source": "aws.rolesanywhere",
  "account": "123456789012",
  "time": "2022-06-10T06:51:08Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:rolesanywhere:us-west-1:123456789012:trust-anchor/61f50cd4-45b9-4259-b049-d0a53682fa4b"
  ],
  "detail": {
    "certificate-serial-number": "00936EACBE07F201DF",
    "days-to-expiry": 3,
    "issuer": "[certificateIssuer]"
  }
}
```

## Intermediate or end-entity certificate expiration event

IAM Roles Anywhere sends an expiration event for intermediate or end-entity certificates when the certificate is within 45 days prior to expiration and used in createSession API. You can use expiration events to configure Amazon SNS to send a text notification whenever IAM Roles Anywhere generates this event.

Expiration events have the following structure.

```
{
  "version": "0",
  "id": "9c95e8e4-96a4-ef3f-b739-b6aa5b193afb",
  "detail-type": "Roles Anywhere Certificate Approaching Expiration Notification",
  "source": "aws.rolesanywhere",
  "account": "123456789012",
  "time": "2022-06-10T06:51:08Z",
  "region": "us-west-1",
  "detail": {
    "certificate-serial-number": "00936EACBE07F201DF",
    "days-to-expiry": 3,
    "issuer": "[certificateIssuer]"
  }
}
```

## Responding to an event

You can configure Amazon Simple Notification Service to send a text notification whenever IAM Roles Anywhere generates an EventBridge event.

### To create an Amazon EventBridge rule that reacts to events

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.

A rule can't have the same name as another rule in the same Region and on the same event bus.
5. For **Event bus**, choose the event bus that you want to associate with this rule. If you want this rule to match events that come from your account, select **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
6. For **Rule type**, choose **Rule with an event pattern**.
7. Choose **Next**.
8. For **Event source**, choose **AWS services**.
9. For **Sample events**, choose an event under **IAM Roles Anywhere**.
10. For **Event pattern**, do the following:
  - a. For **Event source**, choose **AWS services**.
  - b. For **AWS service**, choose **IAM Roles Anywhere**.
  - c. For **Event Type**, choose an **IAM Roles Anywhere** event.
  - d. Choose **Next**
11. In the **Targets** section, choose a service that can consume your event such as Amazon SNS, or choose **Lambda function** to pass the event to customized executable code.

## Logging AWS Identity and Access Management Roles Anywhere API calls using AWS CloudTrail

AWS Identity and Access Management Roles Anywhere is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in IAM Roles Anywhere. CloudTrail captures all API calls for IAM Roles Anywhere as events. The calls captured include calls from the IAM Roles Anywhere console and code calls to the IAM Roles Anywhere API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for IAM Roles Anywhere. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to IAM Roles Anywhere, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

### IAM Roles Anywhere information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in IAM Roles Anywhere, 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 IAM Roles Anywhere, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions 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 further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All IAM Roles Anywhere actions are logged by CloudTrail and are documented in the [IAM Roles Anywhere API Reference](#). For example, calls to the `CreateTrustAnchor`, `ListProfiles`, and `CreateSession` operations generate entries in the CloudTrail log files.

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 AWS Identity and Access Management (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 IAM Roles Anywhere 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 so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `ListProfiles` operation.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAZR5EMTJKE753U4ZDS:test-session",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/test-session",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAZR5EMTJKE753U4ZDS",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2022-03-21T22:40:46Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2022-07-01T18:11:27Z",
```

```
"eventSource": "rolesanywhere.amazonaws.com",
"eventName": "UpdateProfile",
"awsRegion": "us-east-1",
"sourceIPAddress": "1.1.1.1",
"userAgent": "test-agent",
"requestParameters": {
  "durationSeconds": 3600,
  "managedPolicyArns": [
    "arn:aws:iam::aws:policy/AdministratorAccess",
    "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
  ],
  "name": "Updated Test Profile",
  "profileId": "0ace5b12-24b9-427e-a483-c55884852fbf",
  "sessionPolicy": "{\n  \"Version\": \"2012-10-17\", \n  \"Statement\": [\n    {\n      \"Effect\": \"Allow\", \n      \"Action\": \"s3:ListObjects\", \n      \"Resource\": \"*\n    }\n  ]\n}"
},
"responseElements": {
  "profile": {
    "createdAt": "2022-07-01T18:11:27.380711Z",
    "createdBy": "arn:aws:sts::111122223333:assumed-role/Admin/test-session",
    "durationSeconds": 3600,
    "enabled": false,
    "managedPolicyArns": [
      "arn:aws:iam::aws:policy/AdministratorAccess",
      "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    ],
    "name": "Updated Test Profile",
    "profileArn": "arn:aws:rolesanywhere:us-east-1:111122223333:profile/0ace5b12-24b9-427e-a483-c55884852fbf",
    "profileId": "0ace5b12-24b9-427e-a483-c55884852fbf",
    "requireInstanceProperties": false,
    "roleArns": [
      "arn:aws:iam::111122223333:role/test-role"
    ],
    "sessionPolicy": "{\n  \"Version\": \"2012-10-17\", \n  \"Statement\": [\n    {\n      \"Effect\": \"Allow\", \n      \"Action\": \"s3:ListObjects\", \n      \"Resource\": \"*\n    }\n  ]\n}",
    "updatedAt": "2022-07-01T18:11:27.936687Z"
  }
},
"requestID": "ca28860f-504a-4f2d-9f3f-f9cfb4ba0491",
"eventID": "a7bb90c3-c47b-4832-88e7-aeacdda21f1a",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management",
"tlsDetails": {
  "clientProvidedHostHeader": "rolesanywhere.us-east-1.amazonaws.com"
}
}
```

## Monitoring your authentications with AWS Identity and Access Management Roles Anywhere subjects

You can use the **Subject Activity** tab in the IAM Roles Anywhere console to visualize and audit activities for certificates that are authenticated with IAM Roles Anywhere. A *subject* represents a unique identity defined by the X.509 subject of any certificates you use to authenticate with IAM Roles Anywhere. IAM Roles Anywhere creates a subject for you at the time of authentication if there isn't one already for

the X.509 subject. Each subject contains the most recent certificates you have used with IAM Roles Anywhere.

**To view the history of an X.509 subject**

1. Sign in to the [IAM Roles Anywhere console](#).
2. Navigate to the **Subject activity** tab.
3. In the list of certificates records grouped by X.509 **Subject**, choose the **Subject** record that you want to check.
4. On the **Subject details** page, view the details of the subject record.
5. In the **Certificates** section, you can see the most recent record for certificates authenticated with IAM Roles Anywhere that have the same certificate subject.
6. Choose the **Serial number** record to view or copy the certificate body.

# Creating AWS Identity and Access Management Roles Anywhere resources with AWS CloudFormation

AWS Identity and Access Management Roles Anywhere is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as `AWS::RolesAnywhere::Crl`, `AWS::RolesAnywhere::Profile`, and `AWS::RolesAnywhere::TrustAnchor`), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your IAM Roles Anywhere resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

## IAM Roles Anywhere and AWS CloudFormation templates

To provision and configure resources for IAM Roles Anywhere and related services, you must understand [AWS CloudFormation templates](#). Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see [What is AWS CloudFormation Designer?](#) in the *AWS CloudFormation User Guide*.

IAM Roles Anywhere supports creating certificate revocation lists, trust anchors, and profiles in AWS CloudFormation. For more information, including examples of JSON and YAML templates for [CRL](#), [TrustAnchor](#), and [Profile](#), see the *AWS CloudFormation User Guide*.

## Learn more about AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- [AWS CloudFormation](#)
- [AWS CloudFormation User Guide](#)
- [AWS CloudFormation API Reference](#)
- [AWS CloudFormation Command Line Interface User Guide](#)



# AWS Identity and Access Management Roles Anywhere and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Identity and Access Management Roles Anywhere by creating an *interface VPC endpoint*. Interface endpoints are powered by [AWS PrivateLink](#), a technology that enables you to privately access IAM Roles Anywhere APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with IAM Roles Anywhere APIs. Traffic between your VPC and IAM Roles Anywhere does not leave the Amazon network.

Each interface endpoint is represented by one or more [Elastic Network Interfaces](#) in your subnets.

For more information, see [Interface VPC endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.

## Considerations for IAM Roles Anywhere VPC endpoints

Before you set up an interface VPC endpoint for IAM Roles Anywhere, ensure that you review [Interface endpoint properties and limitations](#) in the *Amazon VPC User Guide*.

IAM Roles Anywhere supports making calls to all of its API actions from your VPC.

VPC endpoint policies are supported for IAM Roles Anywhere on all API methods except `CreateSession`. By default, full access to IAM Roles Anywhere is allowed through the endpoint. For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

## Creating an interface VPC endpoint for IAM Roles Anywhere

You can create a VPC endpoint for the IAM Roles Anywhere service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see [Creating an interface endpoint](#) in the *Amazon VPC User Guide*.

Create a VPC endpoint for IAM Roles Anywhere using the following service name:

- `com.amazonaws.region.rolesanywhere`

If you enable private DNS for the endpoint, you can make API requests to IAM Roles Anywhere using its default DNS name for the Region, for example, `rolesanywhere.us-east-1.amazonaws.com`.

For more information, see [Accessing a service through an interface endpoint](#) in the *Amazon VPC User Guide*.

## Creating a VPC endpoint policy for IAM Roles Anywhere

You can attach an endpoint policy to your VPC endpoint that controls access to IAM Roles Anywhere. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

### Example: VPC endpoint policy for IAM Roles Anywhere actions

The following is an example of an endpoint policy for IAM Roles Anywhere. When attached to an endpoint, this policy grants access to the listed IAM Roles Anywhere actions for all principals on all resources.

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "rolesanywhere:CreateTrustAnchor",
        "rolesanywhere:CreateProfile",
        "rolesanywhere:UpdateProfile"
      ],
      "Resource": "*"
    }
  ]
}
```

# Quotas for AWS Identity and Access Management Roles Anywhere

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for AWS Identity and Access Management Roles Anywhere, open the [Service Quotas console](#). In the navigation pane, choose **AWS services** and select **IAM Roles Anywhere**.

To request a quota increase, see [Requesting a Quota Increase](#) in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the [limit increase form](#).

Your AWS account has the following quotas related to IAM Roles Anywhere and each quota is per AWS Region.

Resource	Description	Default value	Adjustable
Combined rate of trust anchor requests	The maximum transactions per second for ListTrustAnchors, CreateTrustAnchor, GetTrustAnchor, UpdateTrustAnchor, DeleteTrustAnchor, EnableTrustAnchor, and DisableTrustAnchor requests combined.	1 per second	Yes
Combined rate of profile requests	The maximum transactions per second for ListProfiles, CreateProfile, GetProfile, UpdateProfile, DeleteProfile, EnableProfile, and DisableProfile requests combined.	1 per second	Yes
Combined rate of subject requests	The maximum transactions per second for ListSubjects and GetSubject requests combined.	1 per second	Yes
Combined rate of tagging requests	The maximum transactions per second for TagResource, UntagResource, and ListTagsForResource requests combined.	1 per second	Yes

Resource	Description	Default value	Adjustable
Combined rate of CRL requests	The maximum transactions per second for ListCrls, GetCrl, ImportCrl, UpdateCrl, DeleteCrl, EnableCrl, and DisableCrl requests combined.	1 per second	Yes
Rate of CreateSession requests	The maximum transactions per second for CreateSession requests.	10 per second	Yes
Trust anchors	The maximum number of trust anchors that you can create within an account.	50	Yes
Profiles	The maximum number of profiles that you can create within an account.	250	Yes
CRLs per trust anchor	The maximum number of Certificate Revocation Lists (CRLs) that you can create per trust anchor within an account.	2	No
Certificates per trust anchor	The maximum number of certificates that you can create per trust anchor within an account.	2	No

## Throttling

Workloads obtain session credentials by using an endpoint that does not use AWS authenticated principals, which would typically be used to limit the rate of operations. IAM Roles Anywhere will limit the rate of calls to the credential endpoint by the authenticating certificate information and IP address (including VPC Endpoint, if applicable).