AWS IAM Anywhere Setup Overview

This document outlines the steps taken to set up AWS IAM Anywhere, enabling on-premises workloads to securely access AWS resources without the need for traditional access keys and secret keys. Instead, AWS IAM Anywhere relies on certificates to authenticate and authorize requests. Below are the steps I followed to configure this system using OpenSSL to generate the required certificates.

This is done using the personal AWS account and Personal Mac OS laptop

---

1. Generating the Certificate Authority (CA)

First, I needed to create a **private CA** that would issue and sign the certificates for the on-premises machines. This private CA serves as the **Trust Anchor** in AWS IAM Anywhere.

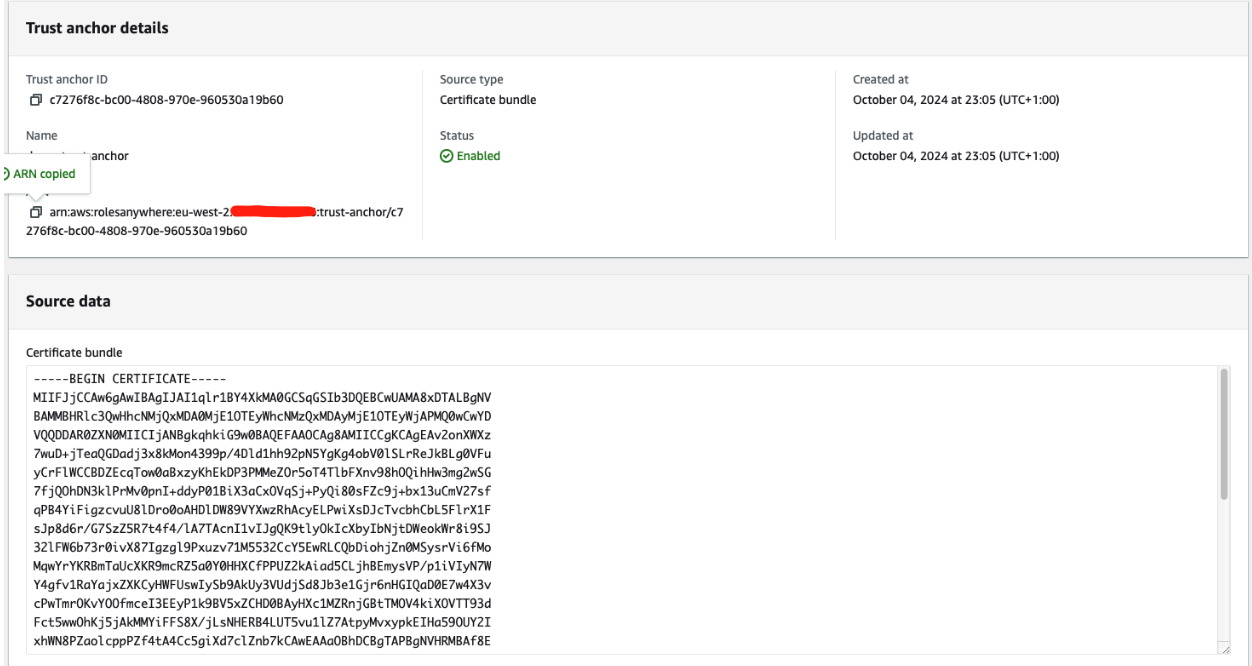
I created the CA's private key and self-signed certificate using OpenSSL:

```
#!/bin/bash

# Create CA private key

openssl genrsa -out MyAWSCA.key 4096



# Create CA certificate with the above-generated private key

openssl req -new -x509 -days 3650 -key MyAWSCA.key -out MyAWSCA.pem
-extensions v3_ca -config ./openssl.cnf
```

- **MyAWSCA.key**: This is the private key for the CA, which is used to sign certificates.
- **MyAWSCA.pem**: This is the self-signed CA certificate, which will act as the **Trust Anchor** in AWS IAM Anywhere.

I uploaded `MyAWSCA.pem` to AWS IAM Anywhere as the **Trust Anchor** so that AWS can trust the certificates issued by this CA.

Create IAM Anywhere Trust Anchor



Trust anchor details

Trust anchor ID
c7276f8c-bc00-4808-970e-960530a19b60

Name
anchor

ARN copied

arn:aws:rolesanywhere:eu-west-2:[redacted]:trust-anchor/c7276f8c-bc00-4808-970e-960530a19b60

Source type
Certificate bundle

Status
Enabled

Created at
October 04, 2024 at 23:05 (UTC+1:00)

Updated at
October 04, 2024 at 23:05 (UTC+1:00)

Source data

Certificate bundle

```
-----BEGIN CERTIFICATE-----
MIIFJjCCAw6gAwIBAgIJAI1qlr1BY4XkMA0GCSqGSIb3DQEBCwUAMA8xDTALBgNV
BAMMBHRlc3QwHhcNMjQxMDA0MjE1OTEyWhcNMzQxMDAyMjE1OTEyWjAPMQ0wCwYD
VQQDDAR0ZXN0MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAv2onXWXz
7wuD+jTeaQGDadj3x8kMon4399p/4Dld1hh92pN5YgKg4obV0lSLrReJkBLg0VFu
yCrFlWCCBDZEcqTow0aBxzyKhEkDP3PMMeZOr5oT4TlbFXnv98hOQihHw3mg2wSG
7fjQOhDN3klPrMv0pnI+ddyP01BiX3aCxOVqSj+PyQi80sFZc9j+bx13uCmV27sf
qPB4YiFigzcvuU8lDro0oAHDlDW89VYXwzRhAcyELPwiXsDJcTvcbhCbL5FlrX1F
sJp8d6r/G7SzZ5R7t4f4/lA7TAcnI1vIJgQK9tlyOkIcXbyIbNjtDWeokWr8i9SJ
32lFW6b73r0ivX87Igzgl9Pxuzv71M5532CcY5EwRLCQbDiohjZn0MSysrVi6fMo
MqwYrYKRBmTaUcXKR9mcRZ5a0Y0HHXCfPPUZ2kAiad5CLjhBEmysVP/p1iVIyN7W
Y4gfv1RaYajxZXKCyHWFUswIySb9AkUy3VUdjSd8Jb3e1Gjr6nHGIQaD0E7w4X3v
cPwTmrOKvYOOfmceI3EEyP1k9BV5xZCHD0BAyHXc1MZRnjGBtTMOV4kiXOVTT93d
Fct5wwOhKj5jAkMMYiFFS8X/jLsNHERB4LUT5vu1lZ7AtpyMvxypkEIHa59OUY2I
xhWN8PZaolcppPZf4tA4Cc5giXd7clZnb7kCAwEAAaOBhDCBgTAPBgNVHRMBAf8E
```

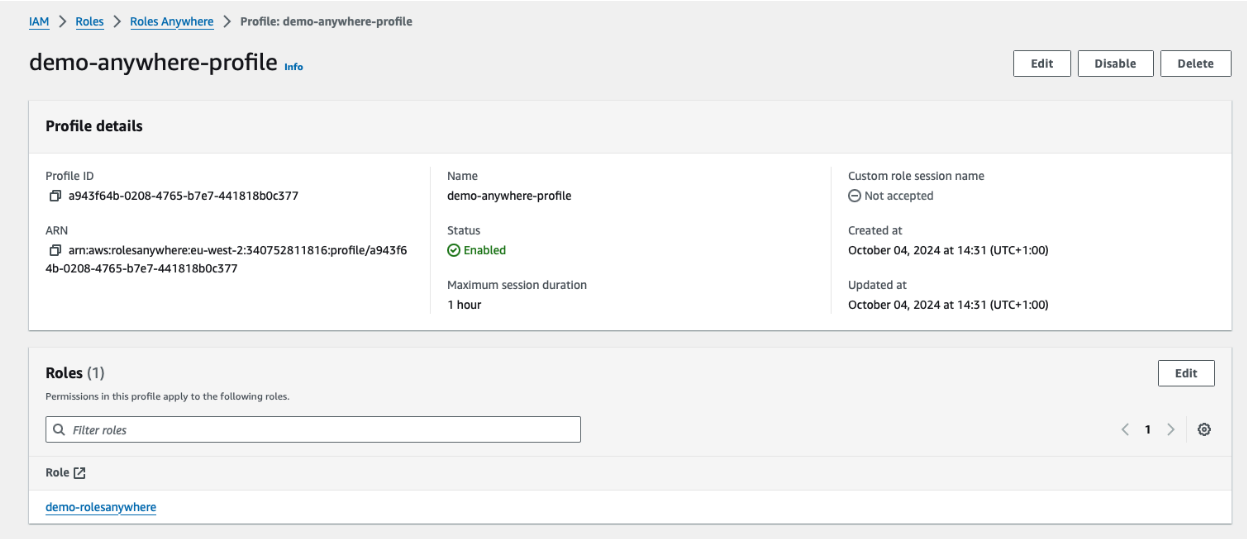
Create IAM Role and Policies

Attach the IAM Policies as required

Use the trust policy

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

Create IAM Anywhere Profile


IAM > Roles > Roles Anywhere > Profile: demo-anywhere-profile
Info
Edit
Disable
Delete
Edit
Filter roles
1

demo-anywhere-profile

Profile details

Profile ID
a943f64b-0208-4765-b7e7-441818b0c377

ARN
arn:aws:rolesanywhere:eu-west-2:340752811816:profile/a943f64b-0208-4765-b7e7-441818b0c377

Name
demo-anywhere-profile

Status
Enabled

Maximum session duration
1 hour

Custom role session name
Not accepted

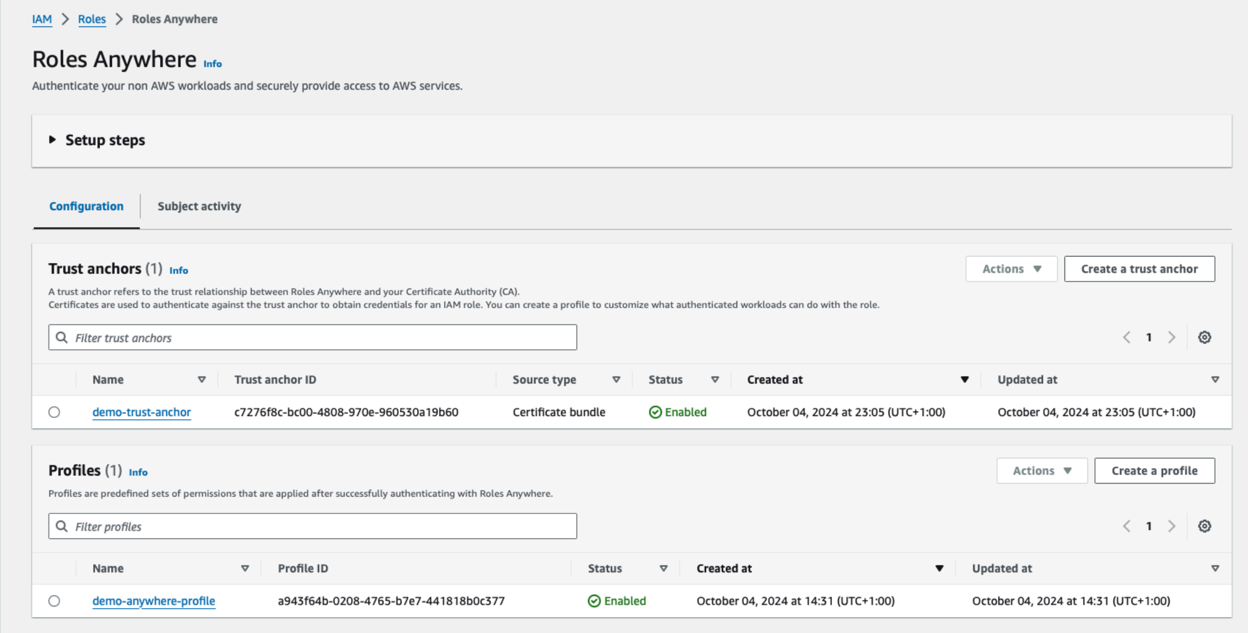
Created at
October 04, 2024 at 14:31 (UTC+1:00)

Updated at
October 04, 2024 at 14:31 (UTC+1:00)

Roles (1)

Permissions in this profile apply to the following roles.

Role
demo-rolesanywhere



Roles Anywhere

Authenticate your non AWS workloads and securely provide access to AWS services.

Trust anchors (1)

A trust anchor refers to the trust relationship between Roles Anywhere and your Certificate Authority (CA).
Certificates are used to authenticate against the trust anchor to obtain credentials for an IAM role. You can create a profile to customize what authenticated workloads can do with the role.

	Name	Trust anchor ID	Source type	Status	Created at	Updated at
○	demo-trust-anchor	c7276f8c-bc00-4808-970e-960530a19b60	Certificate bundle	✓ Enabled	October 04, 2024 at 23:05 (UTC+1:00)	October 04, 2024 at 23:05 (UTC+1:00)

Profiles (1)

Profiles are predefined sets of permissions that are applied after successfully authenticating with Roles Anywhere.

	Name	Profile ID	Status	Created at	Updated at
○	demo-anywhere-profile	a943f64b-0208-4765-b7e7-441818b0c377	✓ Enabled	October 04, 2024 at 14:31 (UTC+1:00)	October 04, 2024 at 14:31 (UTC+1:00)

2. Generating the On-Premises Machine Certificates

Next, I created a certificate for the specific on-premises VM that needs to authenticate with AWS. This process involved creating a private key and a Certificate Signing Request (CSR), then using the private CA to sign the CSR.

```
# Create on-premises private key (specific to the on-premises VM)

openssl genrsa -out onpremise.key 4096


# Create CSR (Certificate Signing Request) for the on-premises VM

openssl req -new -key onpremise.key -out onpremise.csr -config
./openssl.cnf
```

- **onpremise.key**: The private key specific to the on-premises VM, used for encryption and secure communication.
- **onpremise.csr**: The Certificate Signing Request generated using the private key, which will be signed by the CA to issue a valid certificate.

---

3. Signing the On-Premises Certificate Using the CA

Once the CSR was created, I used the CA's private key and certificate (`MyAWSCA.key` and `MyAWSCA.pem`) to sign the on-premises CSR and issue the on-premises certificate.

```
# Create on-premises certificate by signing the CSR with the CA
openssl x509 -req -in onpremise.csr -CA MyAWSCA.pem -CAkey MyAWSCA.key
-CAcreateserial -out onpremise.pem -days 3650 -sha256 -extfile
onpremise.ext
```

- **onpremise.pem**: The signed certificate for the on-premises VM, which will be used to authenticate the machine with AWS.
- **onpremise.ext**: A configuration file defining specific certificate extensions, which I'll explain in detail below.

---

4. The Role of `onpremise.ext`

The `onpremise.ext` file defines important attributes or **extensions** for the certificate. These extensions specify the certificate's intended usage and behavior, ensuring that the certificate is used correctly. Here’s the content of the file:

```
cat > onpremise.ext<<EOF
basicConstraints = CA:FALSE
authorityKeyIdentifier = keyid,issuer
keyUsage = nonRepudiation, digitalSignature, keyEncipherment,
dataEncipherment
EOF
```

- **basicConstraints = CA**
  : This indicates that the certificate is an **end-entity certificate** and not a CA certificate. The on-premises machine cannot issue other certificates, ensuring proper use of roles.
- **authorityKeyIdentifier = keyid,issuer**: This identifies the certificate authority (CA) that issued the certificate, establishing a chain of trust back to the CA.
- **keyUsage = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment**: Specifies what the certificate can be used for, including:
  - **nonRepudiation**: The certificate holder cannot deny the signature.
  - **digitalSignature**: The certificate is used for signing data.
  - **keyEncipherment**: The certificate can be used to encrypt keys.
  - **dataEncipherment**: The certificate can be used to encrypt data directly.

The `onpremise.ext` file is essential as it ensures that the on-premises certificate follows proper security policies. Without these extensions, the certificate might not function as intended, especially in secure environments like AWS IAM Anywhere, where certificate usage must be clearly defined.

---

5. Configure AWS IAM Anywhere Credentials on the On-Premises Machine

1. **Install the AWS IAM Anywhere Helper**:
   - Install the AWS IAM Anywhere signing helper tool on your on-premises machine if not done already. This tool uses the certificate and private key to authenticate requests.
   - https://docs.aws.amazon.com/rolesanywhere/latest/userguide/credential-helper.html
   -
2. **Set Up AWS Credentials**:
   - Configure AWS credentials using the **certificate and private key** on the on-premises machine by updating the `~/.aws/credentials` file:

---

6. Configuring AWS Credentials

Finally, I configured AWS credentials on the on-premises machine to use the certificate for authentication. The AWS `~/.aws/credentials` file was updated with a custom profile, using the client certificate and private key to authenticate via AWS IAM Anywhere:

```
[profile demo-anywhere-profile]
```

```
credential_process = /path/to/aws_signing_helper credential-process
--certificate /path/to/onpremise.pem --private-key
/path/to/onpremise.key --trust-anchor-arn
arn:aws:rolesanywhere:region:account:trust-anchor/123456 --profile-arn
arn:aws:rolesanywhere:region:account:profile/123456 --role-arn
arn:aws:iam::account:role/demo-role
```

In this setup:

- **Trust Anchor**: The CA certificate (`MyAWSCA.pem`) uploaded to AWS.
- **Client Certificate and Private Key**: The certificate (`onpremise.pem`) and key (`onpremise.key`) used by the on-premises VM for authentication.
- **IAM Role**: The IAM role (`demo-role`) that the on-premises VM assumes after successful authentication.

Step 5: Test the Setup

1. **Test Authentication**:
    - Use the configured profile (`demo-anywhere-profile`) to test access to AWS services from your on-premises machine.

For example, run:

```
aws s3 ls --profile demo-anywhere-profile
```

- This command should succeed, proving that the on-premises machine is able to authenticate to AWS using the certificate.

Conclusion

This configuration allows our on-premises workloads to securely access AWS resources using certificates issued by private CA. By leveraging AWS IAM Anywhere and OpenSSL, have created a flexible, secure, and scalable authentication method without relying on access keys and secret keys.

openssl.cnf file used

```
[ req ]
distinguished_name    = req_distinguished_name
attributes      = req_attributes

[ req_distinguished_name ]
countryName                = UK
countryName_min            = 2
countryName_max            = 2
stateOrProvinceName        = Penarth
localityName               = Penarth
0.organizationName         = Penarth
organizationalUnitName           = Penarth
commonName            = test
commonName_max             = 64
emailAddress               = penarth@test.com
emailAddress_max      = 64

[ req_attributes ]
challengePassword          = A challenge password
challengePassword_min      = 4
challengePassword_max      = 20

[ v3_ca ]
basicConstraints        = critical, CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always, issuer:always
keyUsage                = critical, cRLSign, digitalSignature,
keyCertSign
```