OpenSearchCon EUROPE

Enhancing Security in OpenSearch with Dynamic Authentication methods

Sakshi Nasha

Senior Software Engineer

@Cohesity

INTRODUCTION TO YOUR TRULY

OpenSearchCon EUROPE

$whoami

~~Senior s/w Engineer~~ Learner
Community evan·gel·ist (AWS Community Builder)
Innovator : Hackathons
Public speaker
Athlete at heart : 🏃‍♀️🏀⚽🏸
Off the grid? Catch me just soaking up **nature** to recharge for the next big idea.

OpenSearchCon
EUROPE

praise me

Agenda

- Introduction to yours truly 🙋🏻‍♀️
- Why the topic ?
- Static vs Dynamic Creds
- Overview of Dynamic Authentication
- Working of STS Tokens
- Integrating STS with OpenSearch
- Demo time
- SAS token use cases
- Resources
- Q&A

🛡️ Why security in OpenSearch matters

📈 **Increasing Compliance Pressures :**

Data Sovereignty: Ensuring data is stored and accessed in compliance with local laws.

🏢 **Multi-Tenant Environments**

Data Isolation: Preventing unauthorized access between different tenants is so challenging

☁️ **Cloud-Native Deployments**

Dynamic Scaling: Handling the challenges of scaling in cloud environments.

Ephemeral Resources: Managing temporary instances and their security.

Static and Dynamic Creds

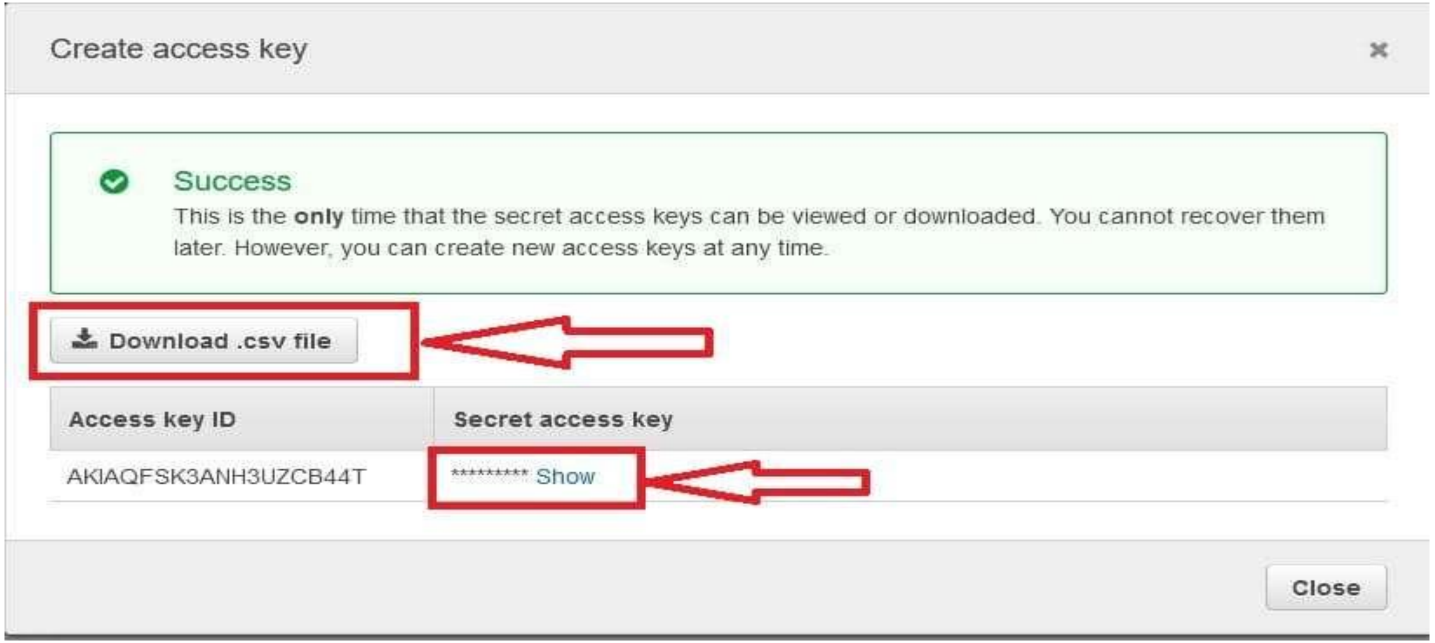
- **Static secrets** are long lived creds
- Such secrets are predefined
- They do not change unless explicitly modified by an administrator or authorized user.

Examples: API keys, database passwords, or encryption keys unless intentionally rotated.

- **Dynamic secrets** have a short lifespan.
- They are typically generated on-demand
- Automatically revoked after a specified time period or usage.

Examples : Temporary Access, Session Token or Short-lived tokens

Traditional ways



Create access key

Success

This is the **only** time that the secret access keys can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time.

Download .csv file

Access key ID	Secret access key
AKIAQFSK3ANH3UZCB44T	********* Show

Close

⚠️ Say no to Static Creds

- Static credentials are **long lived** in nature.
- If compromised, they give attackers **ample time** to exploit the application.
- If they are **stolen** it would be a nightmare to discern which operations are legitimate.
- Thus, the only **fail-safe choice** is to cumbersomely **rotate the keys** and **redistribute to customers**. This is often overlooked action and adds extra pain for the DevOps.

Perks of using Dynamic Creds

1 Enhancing Security

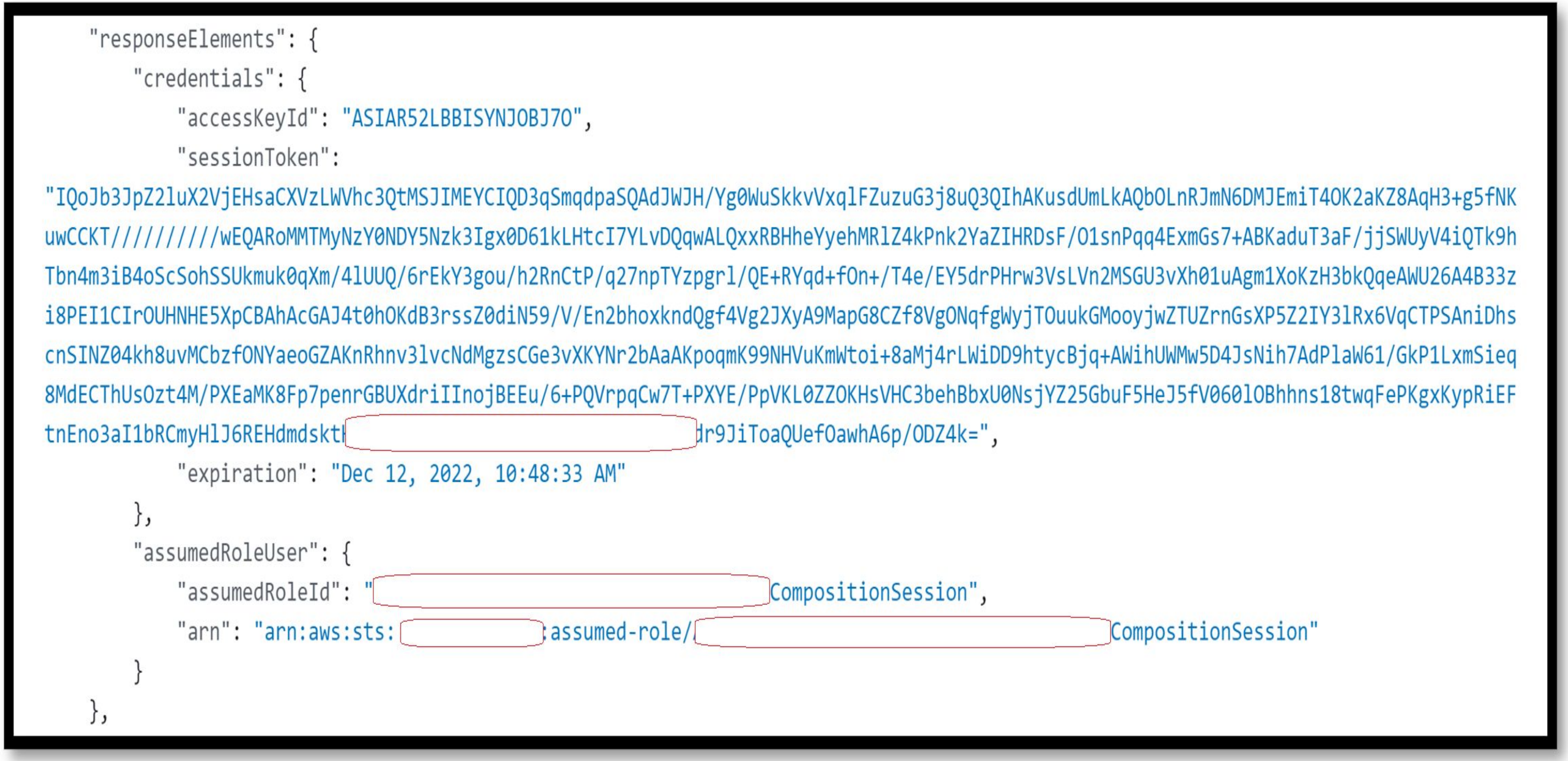
- Short-lived tokens have a **limited lifespan**, reducing the exposure window for potential attacks.
- Validity period is **short**, minimizing the risk of unauthorized access.
- Regular token expiration forces users to **re-authenticate**, ensuring better security.

2 Mitigating Token Abuse

- By making tokens short lived, we limit the time an attacker can use to abuse a stolen token.
- Thus, **minimizing** the **risk window** significantly

3 Least Privilege Principle

- Access to what they absolutely **need** and not what they want.
- When **permissions change** (e.g., user roles or access levels), short-lived tokens **automatically reflect** the updates upon renewal.
- Long-lived tokens may retain outdated permissions, leading to security risks.



```
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAR52LBBISYNJOBJ7O",
            "sessionToken":
"IQoJb3JpZ2luX2VjEHsaCXVzLWVhc3QtMSJIMEYCIQD3qSmqdpaSQAdJWJH/Yg0WuSkkvVxqlFZuzuG3j8uQ3QIhAKusdUmLkAQbOLnRJmN6DMJEmiT4OK2aKZ8AqH3+g5fNK
uwCCKT//////////wEQARoMMTMyNzY0NDY5Nzk3Igx0D61kLHtcI7YLvDQqwALQxxRBHheYyehMRlZ4kPnk2YaZIHRDsF/O1snPqq4ExmGs7+ABKaduT3aF/jjSWUyV4iQTk9h
Tbn4m3iB4oScSohSSUkmuk0qXm/4lUUQ/6rEkY3gou/h2RnCtP/q27npTYzpgrl/QE+RYqd+fOn+/T4e/EY5drPHrw3VsLVn2MSGU3vXh01uAgm1XoKzH3bkQqeAWU26A4B33z
i8PEI1CIrOUHNHE5XpCBAhAcGAJ4t0hOKdB3rssZ0diN59/V/En2bhoxkndQgf4Vg2JXyA9MapG8CZf8VgONqfgWyjTOuukGMooyjwZTUZrnGsXP5Z2IY3lRx6VqCTPSAniDhs
cnSINZ04kh8uvMCbzfONYaeoGZAKnRhnv3lvcNdMgzsCGe3vXKYNr2bAaAKpoqmK99NHVuKmWtoi+8aMj4rLWiDD9htycBjq+AWihUWMw5D4JsNih7AdPlaW61/GkP1LxmSieq
8MdECThUsOzt4M/PXEaMK8Fp7penrGBUXdriIInojBEEu/6+PQVrpqCw7T+PXYE/PpVKL0ZZOKHsVHC3behBbxU0NsjYZ25GbuF5HeJ5fV060lOBhhns18twqFePKgxKypRiEF
tnEno3aI1bRCmyHlJ6REHdmdskt[redacted]dr9JiToaQUefOawhA6p/ODZ4k=",
            "expiration": "Dec 12, 2022, 10:48:33 AM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "[redacted]CompositionSession",
            "arn": "arn:aws:sts:[redacted]:assumed-role/[redacted]CompositionSession"
        }
    },
```

I hope I have convinced you enough to switch to Dynamic ways to Authenticate !!

Dynamic Creds in OpenSearch

- The Security plugin in OpenSearch allows you to configure two types of authentication tokens:
  - **On-Behalf-Of (OBO) tokens**
  - **Service Account tokens.**

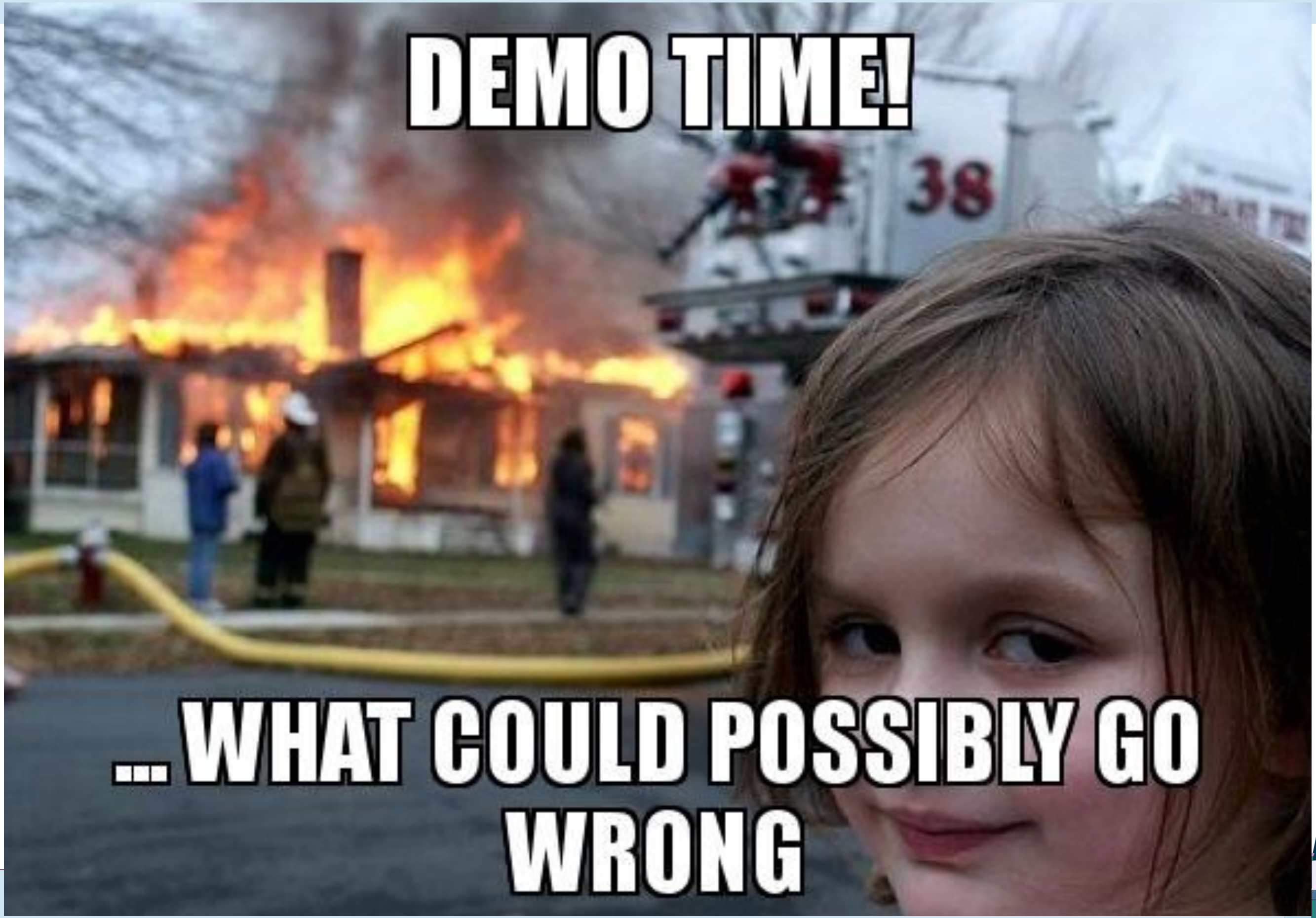
🔐 On-Behalf-Of (OBO) Tokens in OpenSearch

- **Purpose**:
  - Allow extensions or services to perform actions **on behalf of a user**, inheriting the user's privileges for a short duration.
- **Token Type:** JSON Web Token (JWT)
- **Validity**: Configurable window of validity
- **Usage**: These tokens operate "just-in-time," i.e.Issued immediately before required for authentication
- **Configuration**: Defined in the OpenSearch config.yml file under the dynamic configuration section

🛠️ Service Account Tokens in OpenSearch

OpenSearchCon EUROPE

- **Purpose**:
  - Allow extensions to perform actions autonomously, without assuming the role(s) of the active user.
- **Token Type:** Service Account Token
- **Validity**: Configurable window of validity
- **Usage**: Designed for system-level operations
- **Configuration**:Defined in the OpenSearch config.yml file under the service account section



Wish me luck

Demo Walkthrough

1) Create **Opensearch domain** with IAM as master

2) Create **IAMUser** : OpenSearchDemoUser1

3) Create **IAM Policy** : OpenSearchDemoPolicy

4) Attach a IAM policy" OpenSearchDemoPolicy" to the user which allows access to your domain of OpenSearch

5) Create **IAM Role** : OpenSearchDemoRole

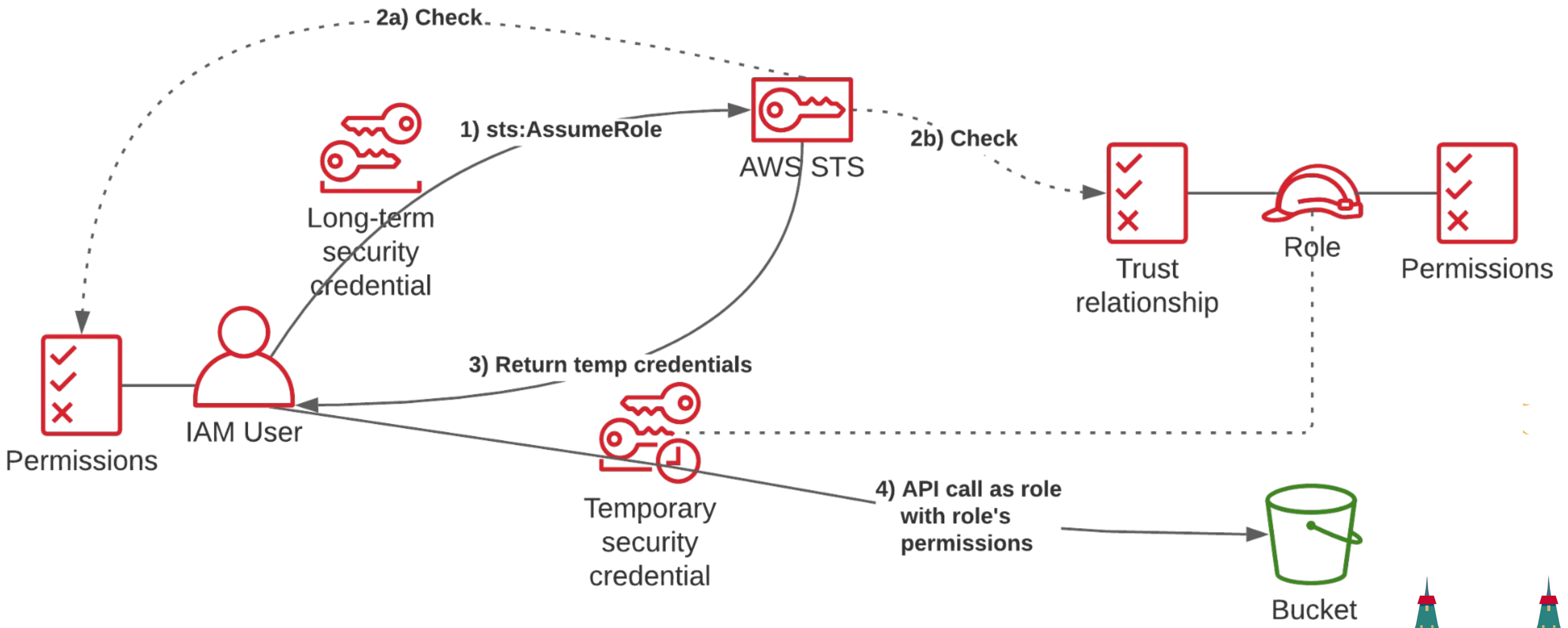
6) Attach Permissions to role

- AmazonOpenSearchServiceFullAccess : AWS Managed
- And OpenSearchDemoPolicy : Customer managed

7) Set up the **trust relationship** in the role to allow the user OpenSearchDemoUser1 to assume the role

Dynamic Creds in OpenSearch

OpenSearchCon EUROPE


2a) Check
1) sts:AssumeRole
AWS STS
2b) Check
Long-term security credential
Trust relationship
Role
Permissions
Permissions
IAM User
3) Return temp credentials
Temporary security credential
4) API call as role with role's permissions
Bucket

Dynamic Creds in OpenSearch

Component	**Role in your setup**
IAM User (OpenSearchDemoUser1)	The *user being represented* in OBO model
IAM Role (OpenSearchDemoRole)	The *permissions* the user temporarily assumes
STS:AssumeRole	The *mechanism* to get temporary credentials "on behalf of" the user
OpenSearch	The *service* being accessed with those temporary credentials

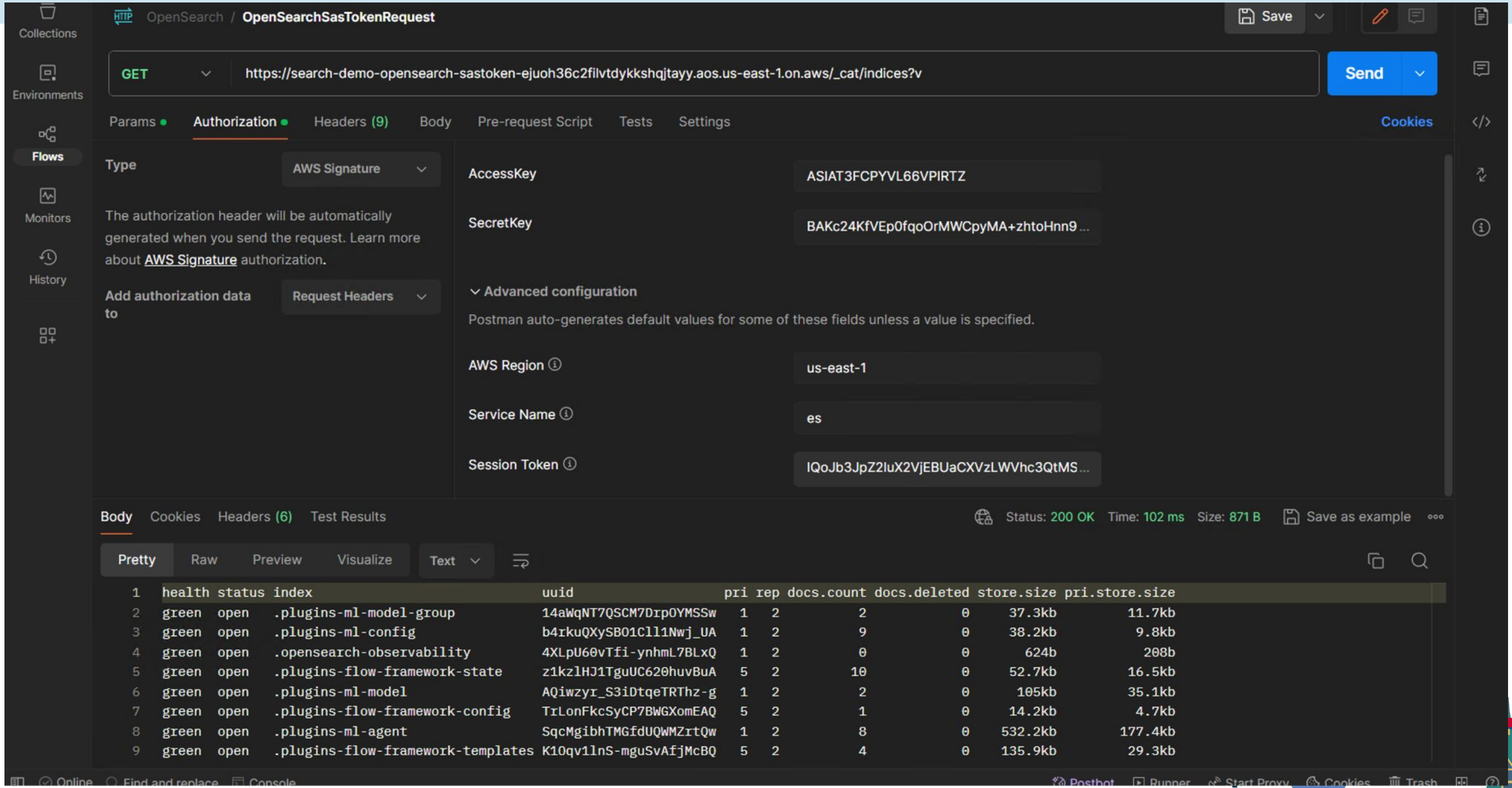
Demo Walkthrough

1. Use **AWS CLI** to create sts

    ```
    aws sts assume-role --role-arn
    arn:aws:iam::264481260887:role/OpenSearchDemoRole
    --role-session-name OpenSearchSession --profile OpenSearchDemoUser1
    --duration-seconds 900
    ```

2. **Hit the endpoint** with STS token thru Postman

    ```
    GET
    https://search-demo-opensearch-sastoken-ejuoh36c2filvtdykkshqjtayy.aos.us-east-1.on.aws/_cat/indices?v
    ```



GET https://search-demo-opensearch-sastoken-ejuoh36c2filvtdykkshqjtayy.aos.us-east-1.on.aws/_cat/indices?v

Type: AWS Signature

The authorization header will be automatically generated when you send the request. Learn more about AWS Signature authorization.

Add authorization data to: Request Headers

AccessKey: ASIAT3FCPYVL66VPIRTZ

SecretKey: BAKc24KfVEp0fqoOrMWCpyMA+zhtoHnn9...

Advanced configuration

Postman auto-generates default values for some of these fields unless a value is specified.

AWS Region: us-east-1

Service Name: es

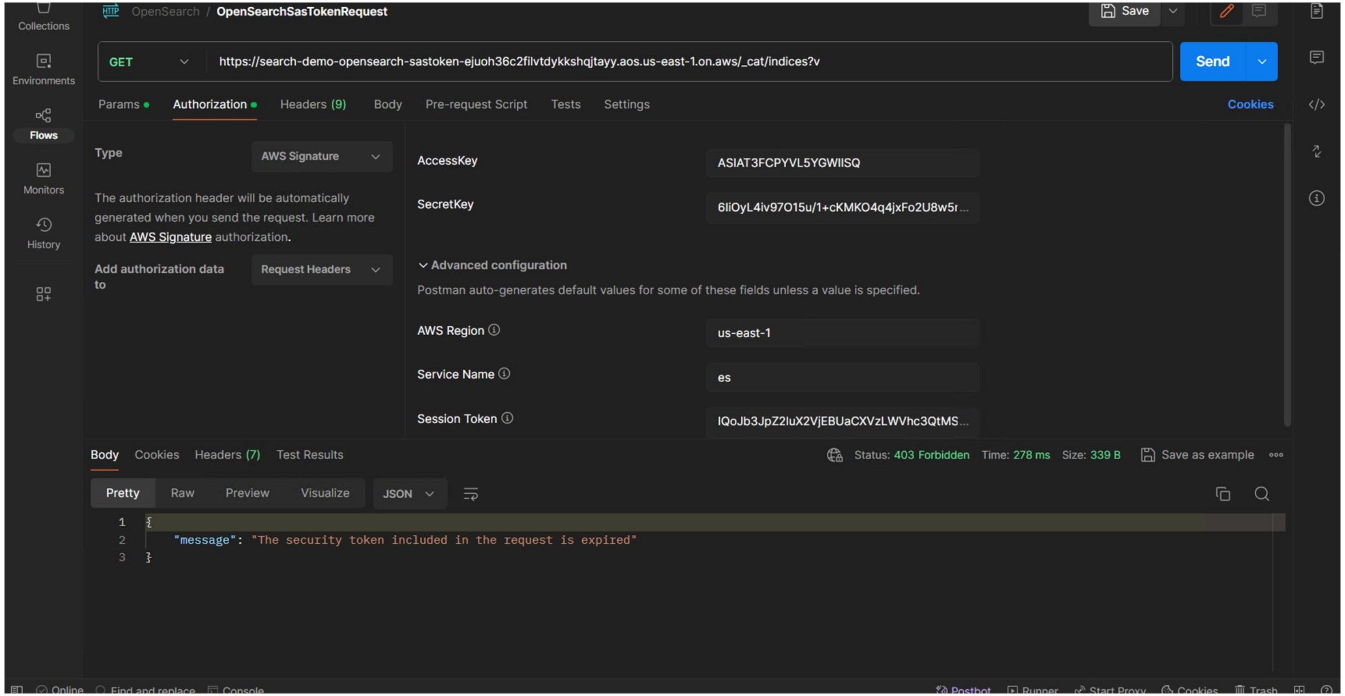
Session Token: IQoJb3JpZ2luX2VjEBUaCXVzLWVhc3QtMS...

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 102 ms Size: 871 B

```
health status index                            uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   .plugins-ml-model-group          14aWqNT7QSCM7DrpOYMSSw   1   2          2            0     37.3kb         11.7kb
green  open   .plugins-ml-config               b4rkuQXySBO1Cll1Nwj_UA   1   2          9            0     38.2kb          9.8kb
green  open   .opensearch-observability        4XLpU60vTfi-ynhmL7BLxQ   1   2          0            0       624b           208b
green  open   .plugins-flow-framework-state    z1kzlHJ1TguUC620huvBuA   5   2         10            0     52.7kb         16.5kb
green  open   .plugins-ml-model                AQiwzyr_S3iDtqeTRThz-g   1   2          2            0      105kb         35.1kb
green  open   .plugins-flow-framework-config   TrLonFkcSyCP7BWGXomEAQ   5   2          1            0     14.2kb          4.7kb
green  open   .plugins-ml-agent                SqcMgibhTMGfdUQWMZrtQw   1   2          8            0    532.2kb        177.4kb
green  open   .plugins-flow-framework-templates K1Oqv1lnS-mguSvAfjMcBQ   5   2          4            0    135.9kb         29.3kb
```

After 15 mins



GET https://search-demo-opensearch-sastoken-ejuoh36c2filvtdykkshqjtayy.aos.us-east-1.on.aws/_cat/indices?v

Type: AWS Signature

The authorization header will be automatically generated when you send the request. Learn more about AWS Signature authorization.

Add authorization data to: Request Headers

AccessKey: ASIAT3FCPYVL5YGWIISQ

SecretKey: 6liOyL4iv97O15u/1+cKMKO4q4jxFo2U8w5r...

Advanced configuration

Postman auto-generates default values for some of these fields unless a value is specified.

AWS Region: us-east-1

Service Name: es

Session Token: IQoJb3JpZ2luX2VjEBUaCXVzLWVhc3QtMS...

Body | Cookies | Headers (7) | Test Results

Status: 403 Forbidden Time: 278 ms Size: 339 B Save as example

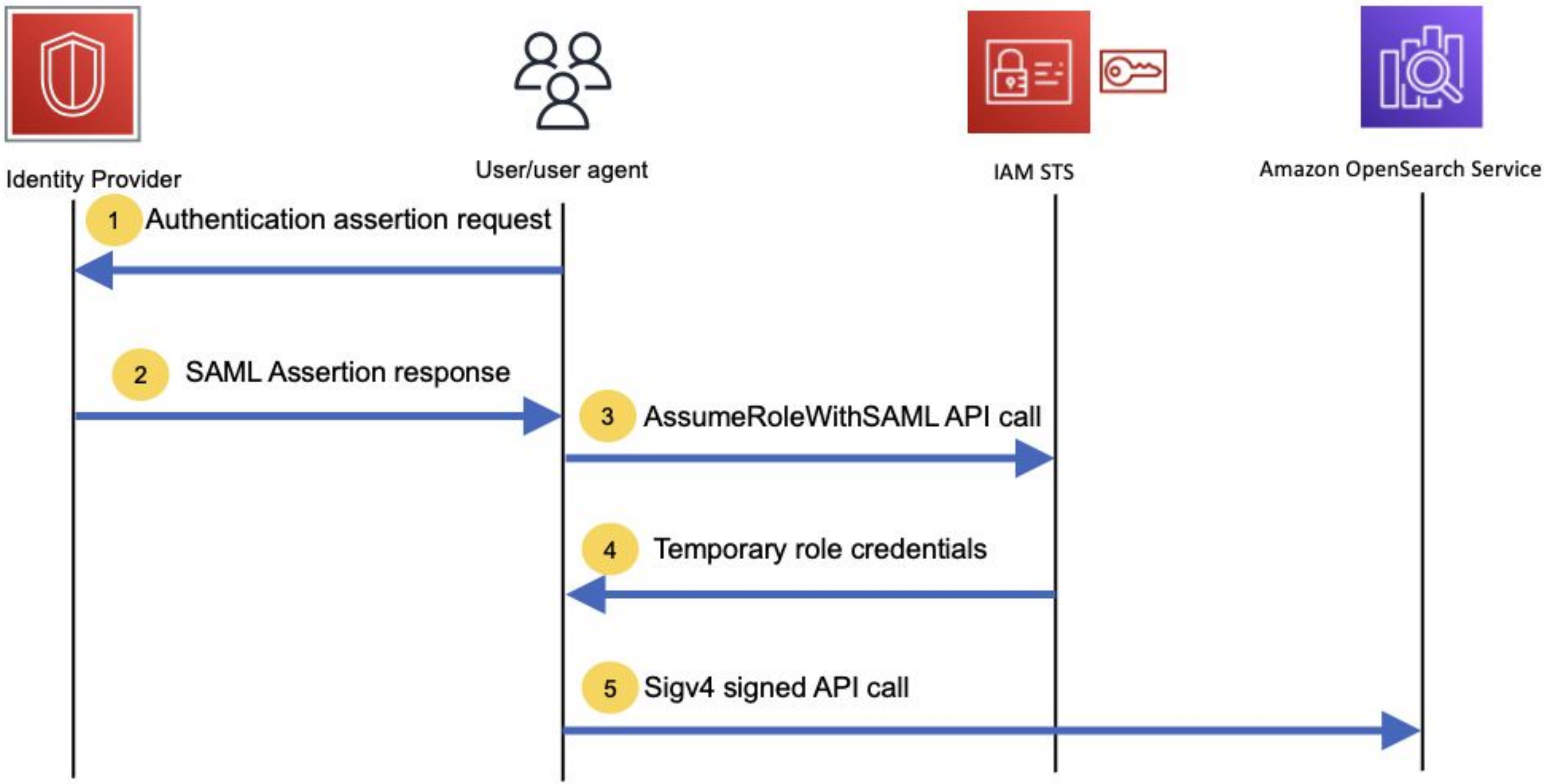
Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "The security token included in the request is expired"
}
```

SAS (Shared Access Signatures) Use Case

🔐 1. Secure Web App Access (Temporary User)

- Scenario: A user logs into a web application, and you want to allow them to view their own data in OpenSearch for the next 30 minutes.
- How it works:
  - User authenticates with Cognito or Identity Provider (IdP)
  - Backend calls AssumeRoleWithWebIdentity or GetSessionToken to get STS credentials
  - Backend uses these credentials with Postman (or SDKs) to query OpenSearch
  - Access is scoped to specific index or document type via IAM/OpenSearch role mapping
- 🔒 Ideal for dashboards, analytics, and search in a multi-tenant SaaS platform.

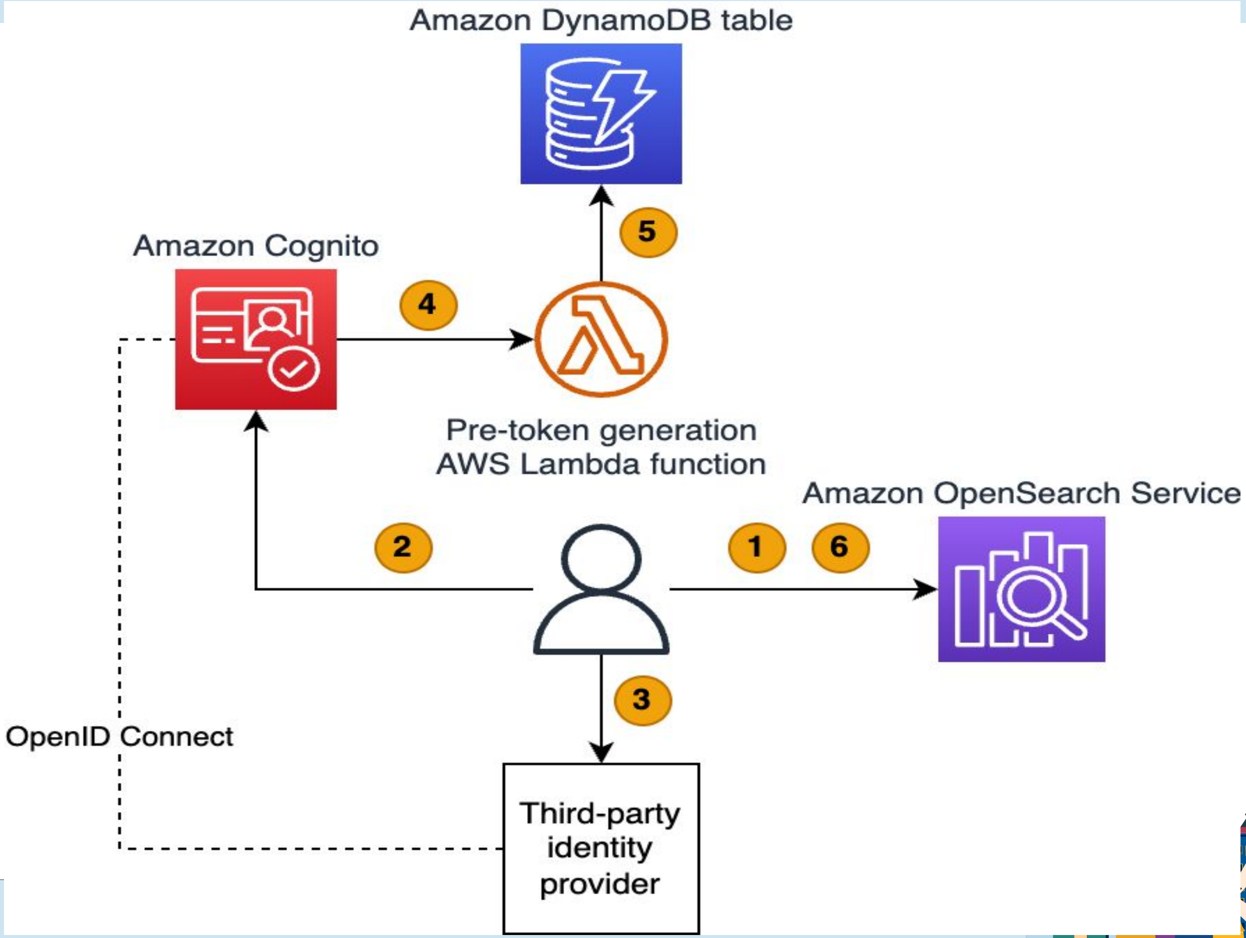

Identity Provider
User/user agent
IAM STS
Amazon OpenSearch Service
1 Authentication assertion request
2 SAML Assertion response
3 AssumeRoleWithSAML API call
4 Temporary role credentials
5 Sigv4 signed API call

SAS (Shared Access Signatures) Use Case

🧾 2. Run Time-Limited Data Ingest Jobs

- Scenario: A scheduled Lambda function pushes data to OpenSearch for 1 hour each night.
- How it works:
  - Lambda assumes a role with AssumeRole and gets temporary credentials
  - Uses these to push logs or metrics into OpenSearch
  - Access expires after job ends (e.g. 1 hour)
- 🔄 Great for ETL pipelines and log processing jobs.

SAS (Shared Access Signatures) Use Case

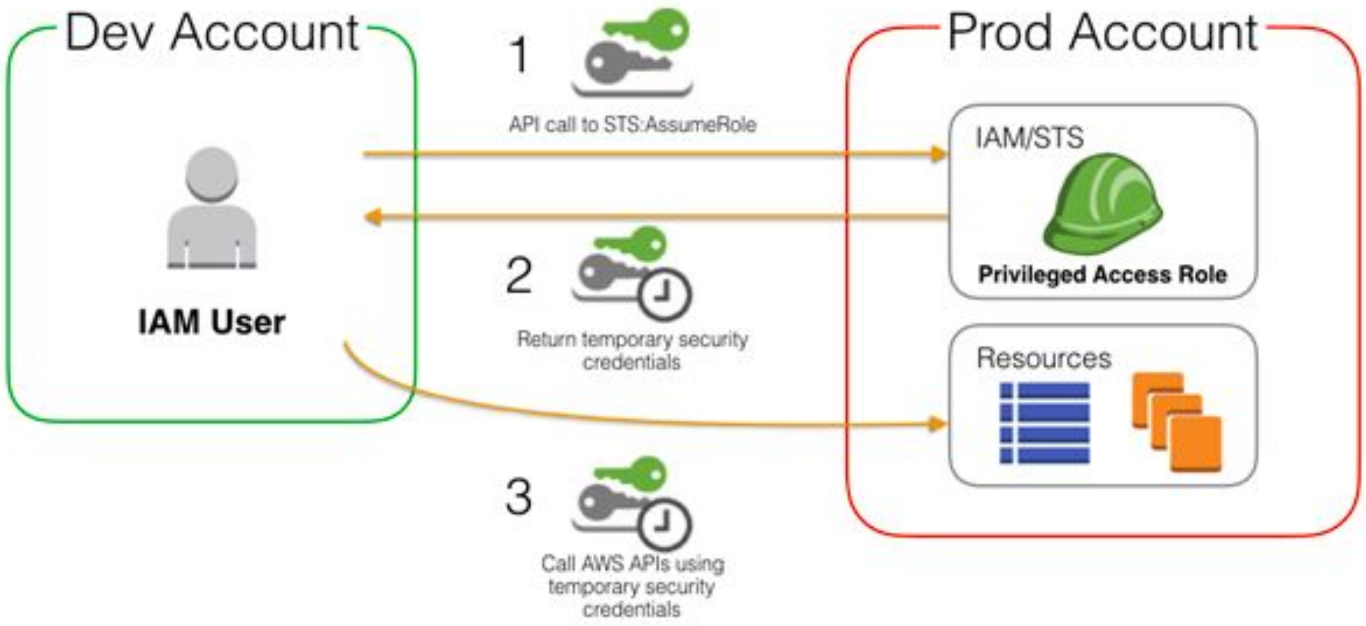


SAS (Shared Access Signatures) Use Case

🔁 3. Cross-Account Access

- Scenario: Your OpenSearch domain is in Account A, and a script running in Account B needs to query it.
- How it works:
  - Account B assumes a role in Account A using STS
  - Gets credentials valid for 1 hour
  - Uses those credentials to access OpenSearch securely
- 🧩 Useful in multi-account AWS orgs or vendor integrations

SAS (Shared Access Signatures) Use Case


Dev Account
IAM User
1
API call to STS:AssumeRole
2
Return temporary security credentials
3
Call AWS APIs using temporary security credentials
Prod Account
IAM/STS
Privileged Access Role
Resources

To know more on Service Account

Documentation

OpenSearchCon EUROPE

SCAN ME

Connect with me

OpenSearchCon EUROPE

Thank you for being an amazing audience

That’s how I want the hall to look like !!!

BEDANKT | Thank you

Until Next time …

OpenSearchCon EUROPE

Q&A, Bring it on !!