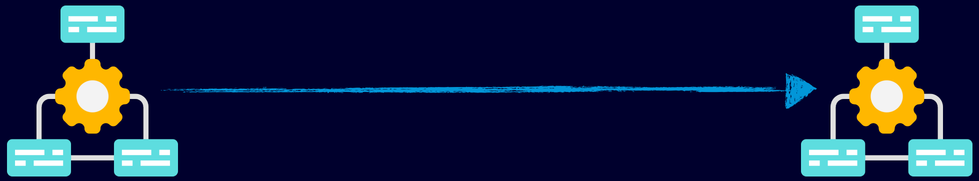
Service-To-Service Authentication And Authorization Using AWS SigV4

Sebastian Anton
DevOps Engineer
NEP Norway

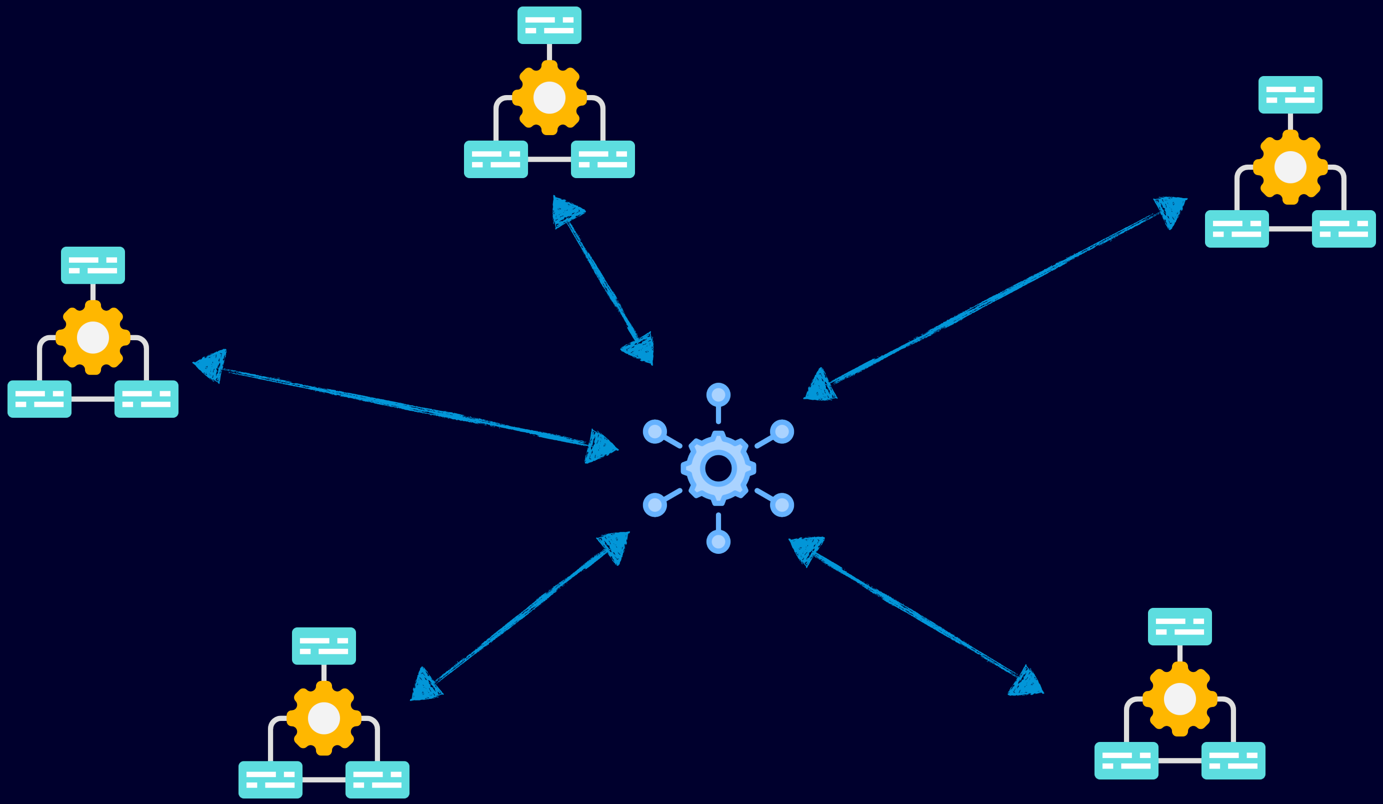
NEPGROUP.COM

Service-to-Service Authentication

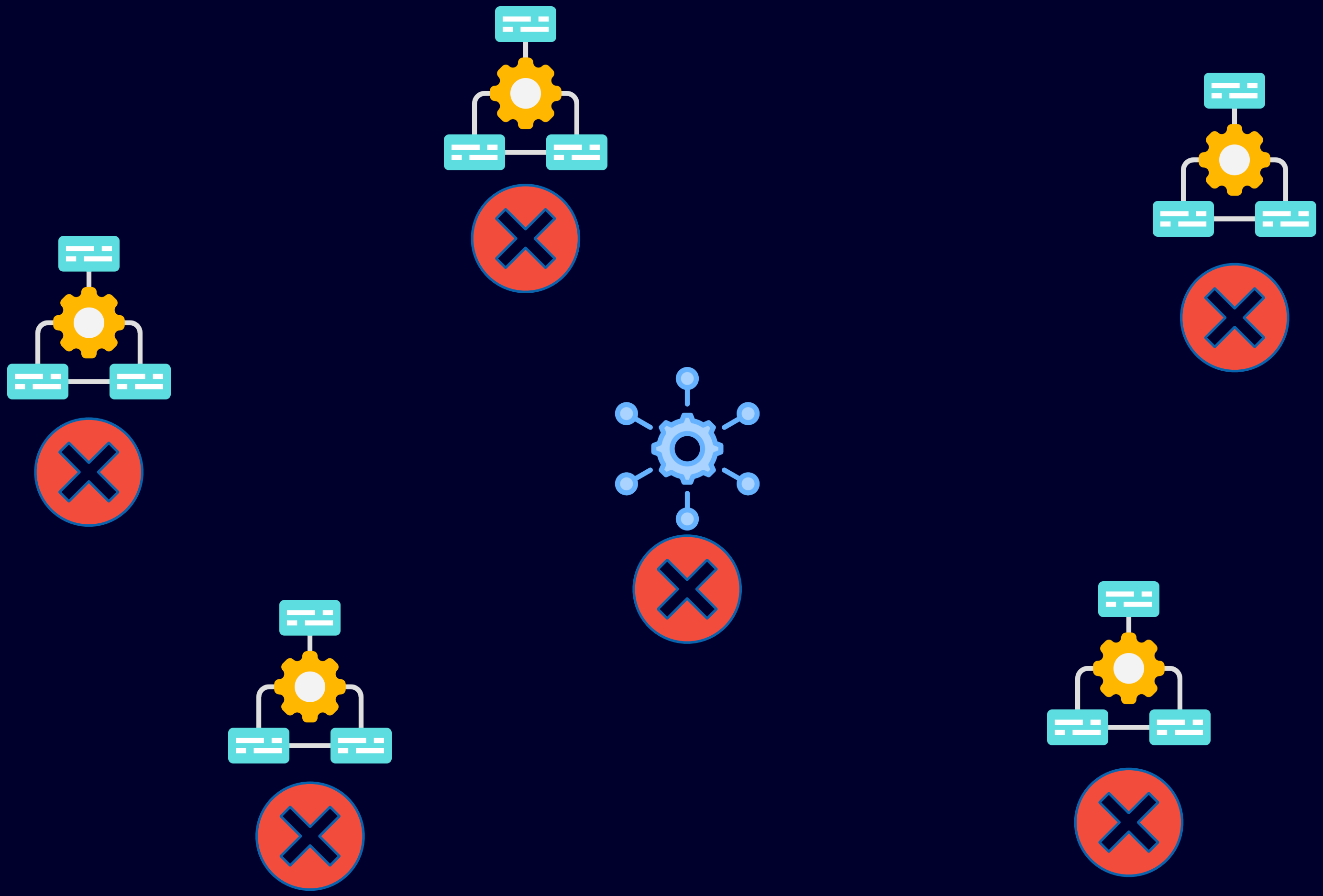


Do I know you?

Service-to-Service Authentication



Service-to-Service Authentication



How Is AWS Securing their APIs?

AWS uses SigV4 signed request

Can We Do the Same?

Yes, we need:

- API Gateway in front of the called service
- AWS SigV4 signed request from the caller service

AWS Signature Version 4 (SigV4)

What is SigV4?

- **AWS's implementation of *HMAC Over HTTP(S)* for signing requests. (Hash-Based Message Authentication Code)**

- **Used to authenticate requests to AWS services (like S3 or API Gateway).**

- **Prevents unauthorized access and tampering by validating the authenticity of the request.**

- **Already added by default in AWS CLI and SDK calls**

SigV4 Signing Process

Steps:

Create a canonical request

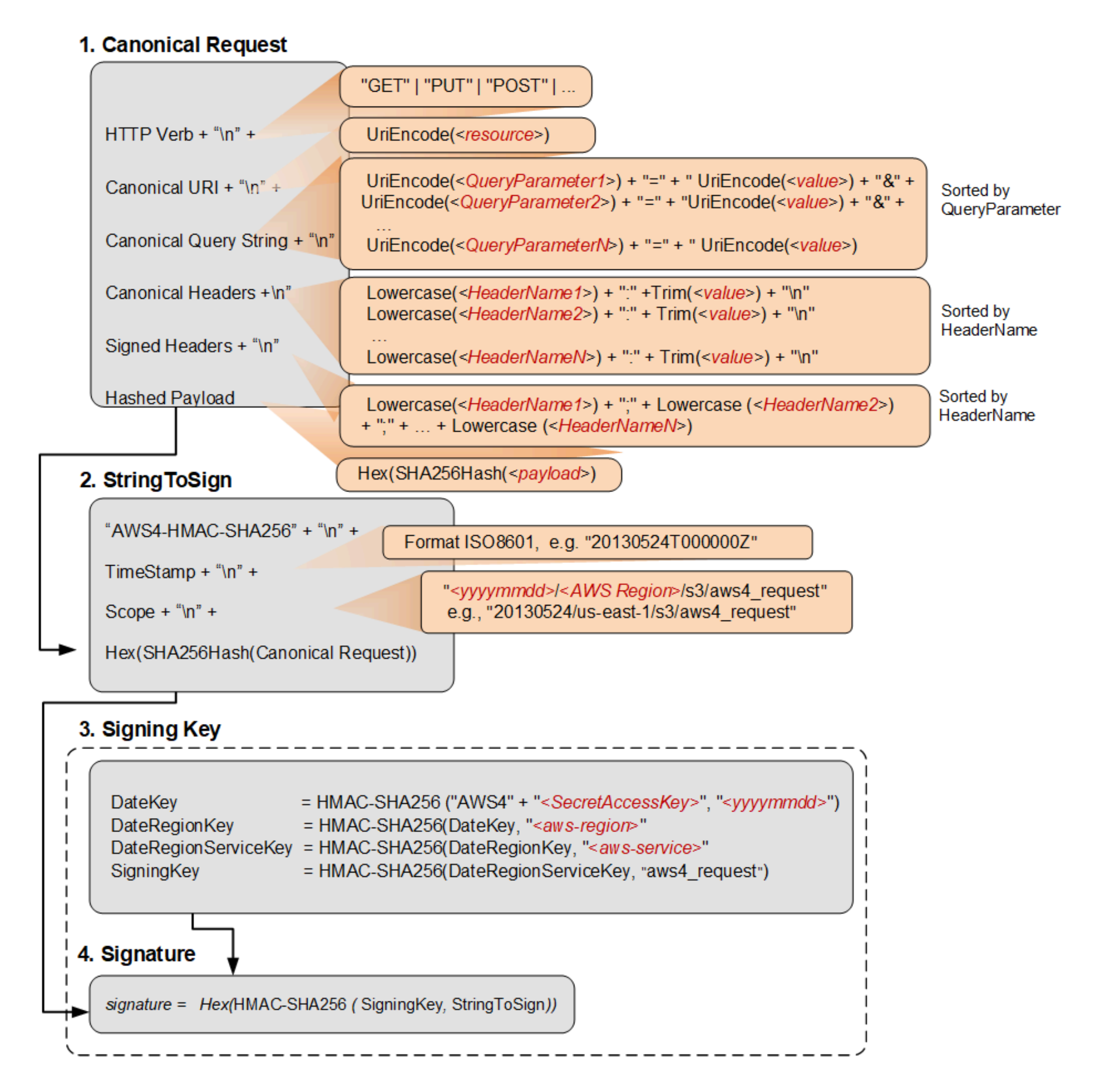
Create a hash of the canonical request

Create a String to Sign

Derive a signing key

Calculate the signature

Add the signature to the request



1. Canonical Request

HTTP Verb + “\n” +

Canonical URI + “\n” +

Canonical Query String + “\n”

Canonical Headers +\n”

Signed Headers + “\n”

Hashed Payload

"GET" | "PUT" | "POST" | ...

UriEncode(<*resource*>)

UriEncode(<*QueryParameter1*>) + "=" + " UriEncode(<*value*>) + "&" +
UriEncode(<*QueryParameter2*>) + "=" + "UriEncode(<*value*>) + "&" +
...
UriEncode(<*QueryParameterN*>) + "=" + " UriEncode(<*value*>)

Sorted by QueryParameter

Lowercase(<*HeaderName1*>) + ":" +Trim(<*value*>) + "\n"
Lowercase(<*HeaderName2*>) + ":" + Trim(<*value*>) + "\n"
...
Lowercase(<*HeaderNameN*>) + ":" + Trim(<*value*>) + "\n"

Sorted by HeaderName

Lowercase(<*HeaderName1*>) + ";" + Lowercase (<*HeaderName2*>)
+ ";" + ... + Lowercase (<*HeaderNameN*>)

Sorted by HeaderName

Hex(SHA256Hash(<*payload*>)

2. StringToSign

“AWS4-HMAC-SHA256” + “\n” +

TimeStamp + “\n” +

Scope + “\n” +

Hex(SHA256Hash(Canonical Request))

Format ISO8601, e.g. "20130524T000000Z"

"<*yyyymmdd*>/<*AWS Region*>/s3/aws4_request"
e.g., "20130524/us-east-1/s3/aws4_request"

3. Signing Key

DateKey = HMAC-SHA256 ("AWS4" + "<*SecretAccessKey*>", "<*yyyymmdd*>")
DateRegionKey = HMAC-SHA256(DateKey, "<*aws-region*>"
DateRegionServiceKey = HMAC-SHA256(DateRegionKey, "<*aws-service*>"
SigningKey = HMAC-SHA256(DateRegionServiceKey, "aws4_request")

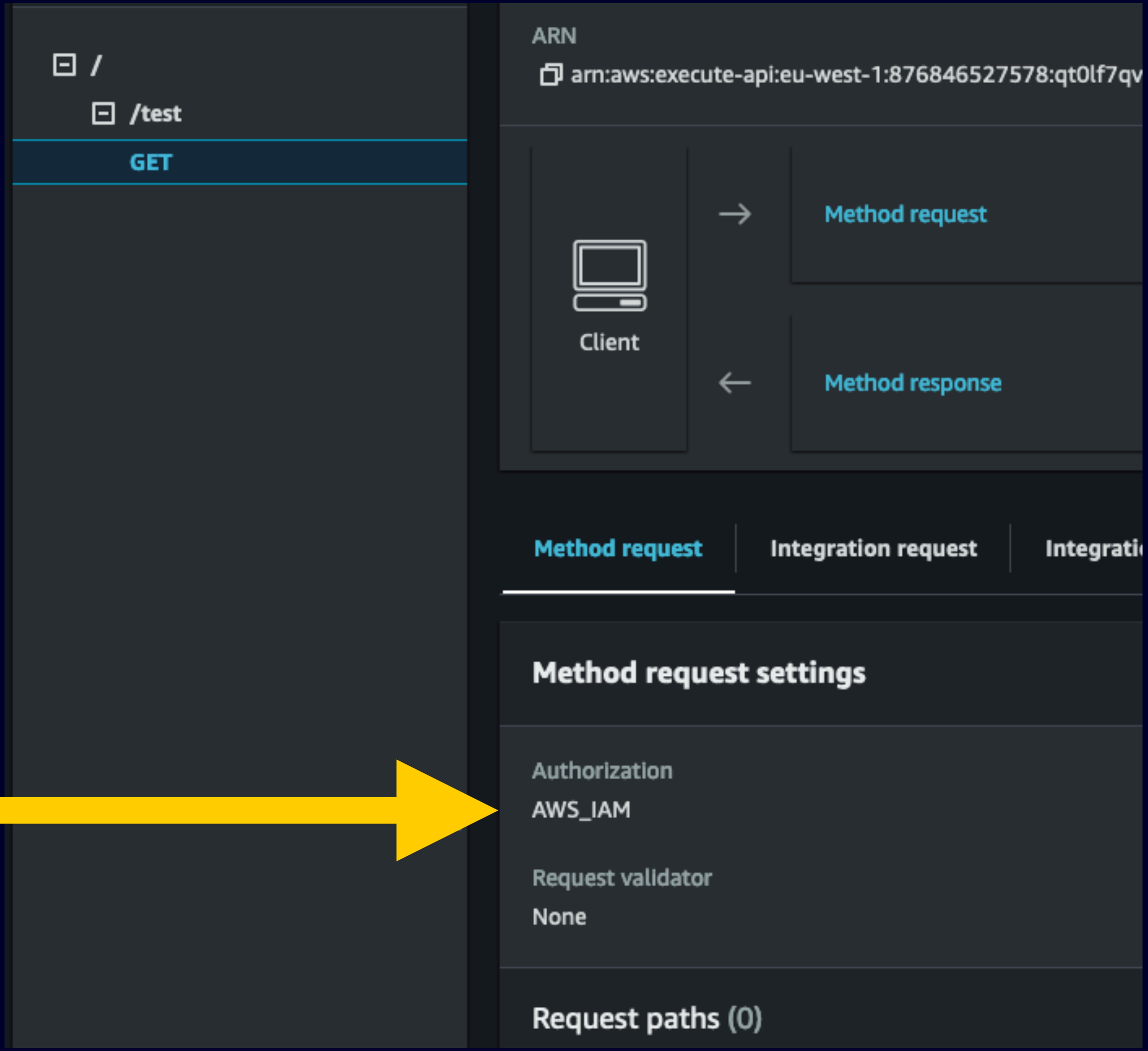
4. Signature

signature = *Hex(*HMAC-SHA256 *(* SigningKey, StringToSign*))*

API Gateway with IAM Authorizer

The authorizationType needs to be set to AWS_IAM for the method.

Required for API Gateway to not skip the signature check



ARN
arn:aws:execute-api:eu-west-1:876846527578:qt0lf7qv

Method request settings

Authorization
AWS_IAM

Request validator
None

Request paths (0)

API Gateway Resource Policies

Example to allow cross account access.

This delegates the permissions to the specified account.

Used when assume role is not desired

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": “arn:aws:iam::111111111111:root"
      },
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:eu-west-1:111111111111:Api-Id/*"
    },
  ]
}
```

Resource format:

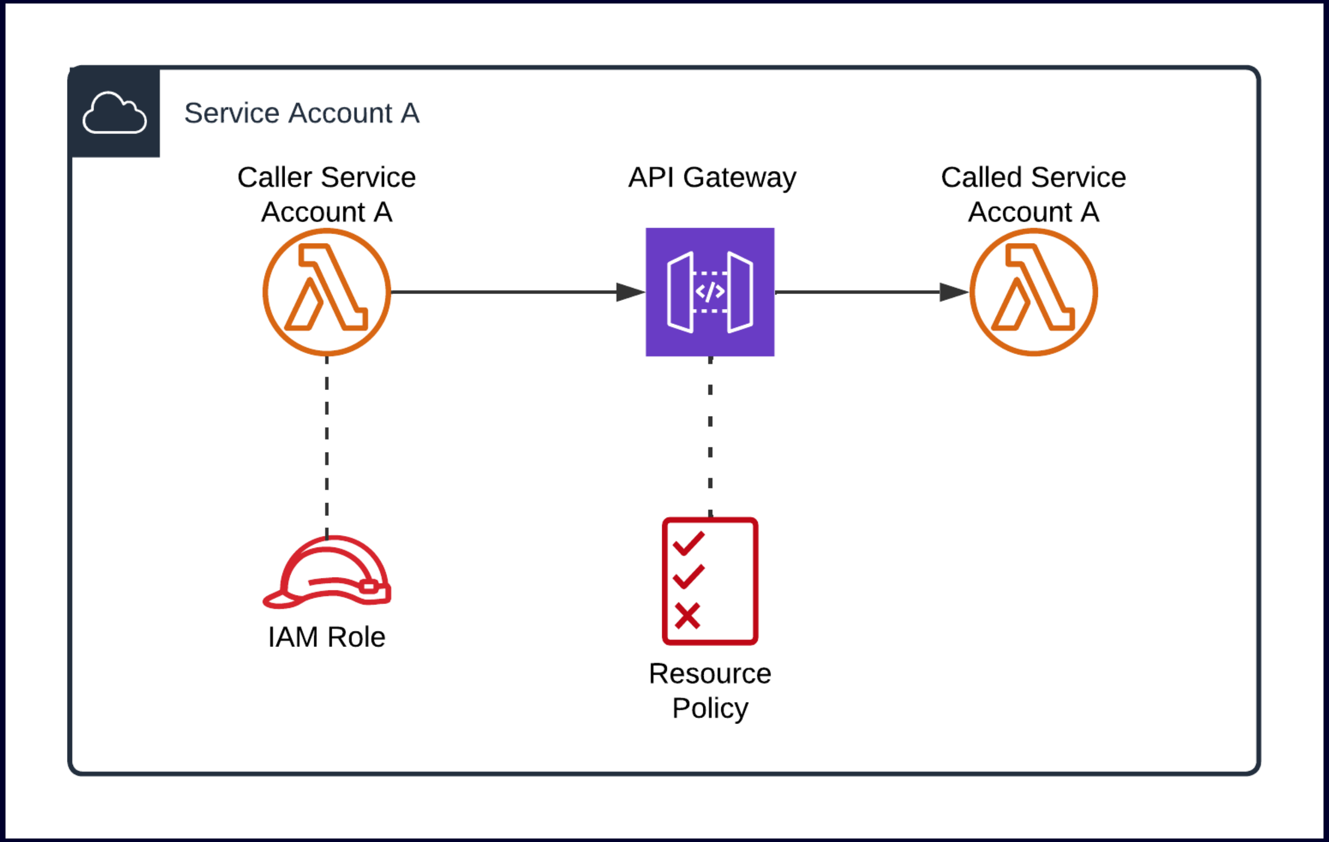
```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/path
```

Signing Requests in your Service

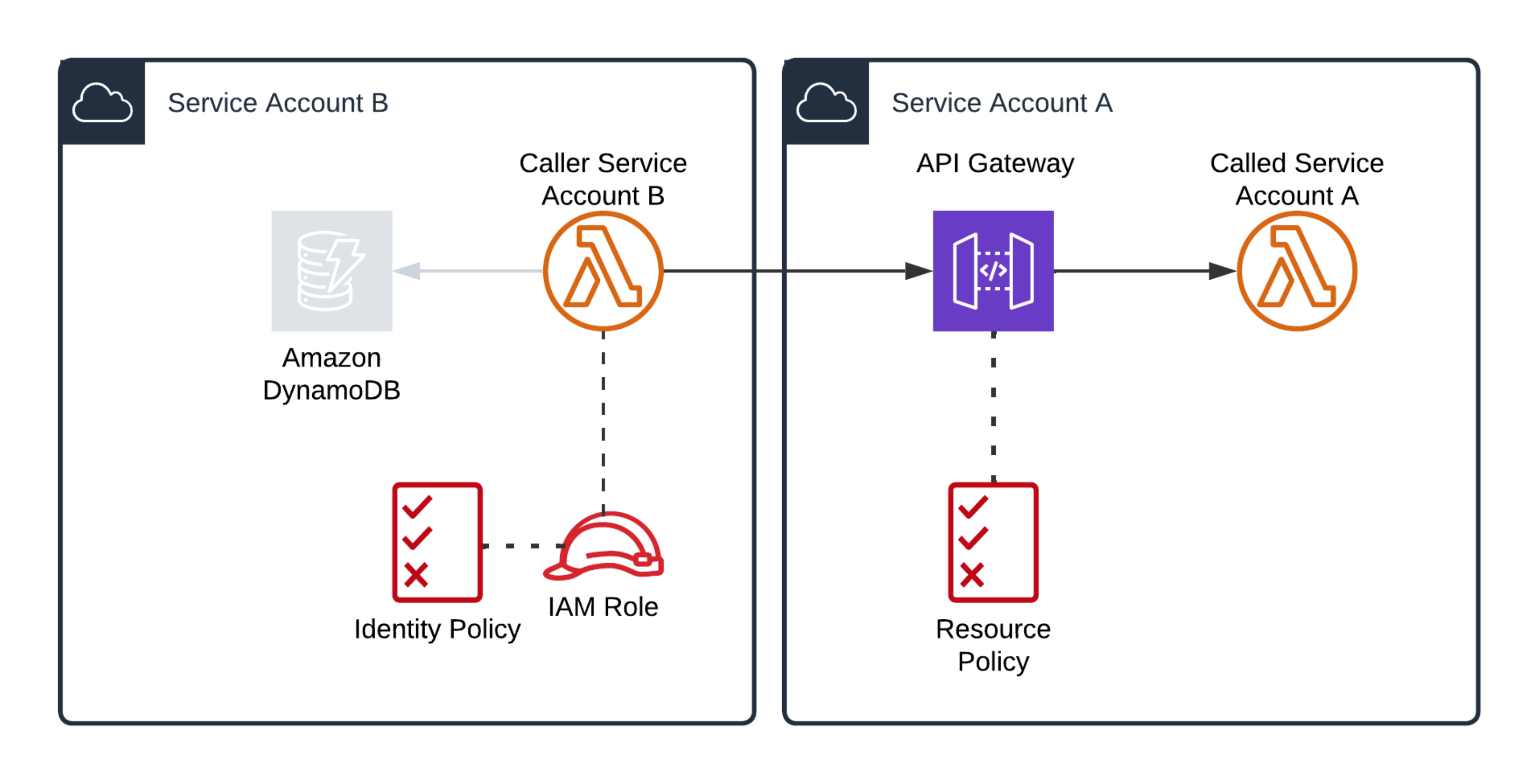
Example using Axios and aws4-axios interceptor in NodeJS

```
import axios from "axios";
import { aws4Interceptor } from "aws4-axios";

const interceptor = aws4Interceptor({
  options: {
    region: "eu-west-2",
    service: "execute-api",
  },
});

axios.interceptors.request.use(interceptor);

// Requests made using Axios will now be signed
axios.get("https://example.com/foo").then((res) => {
  // ...
});
```

Demo: Same Account

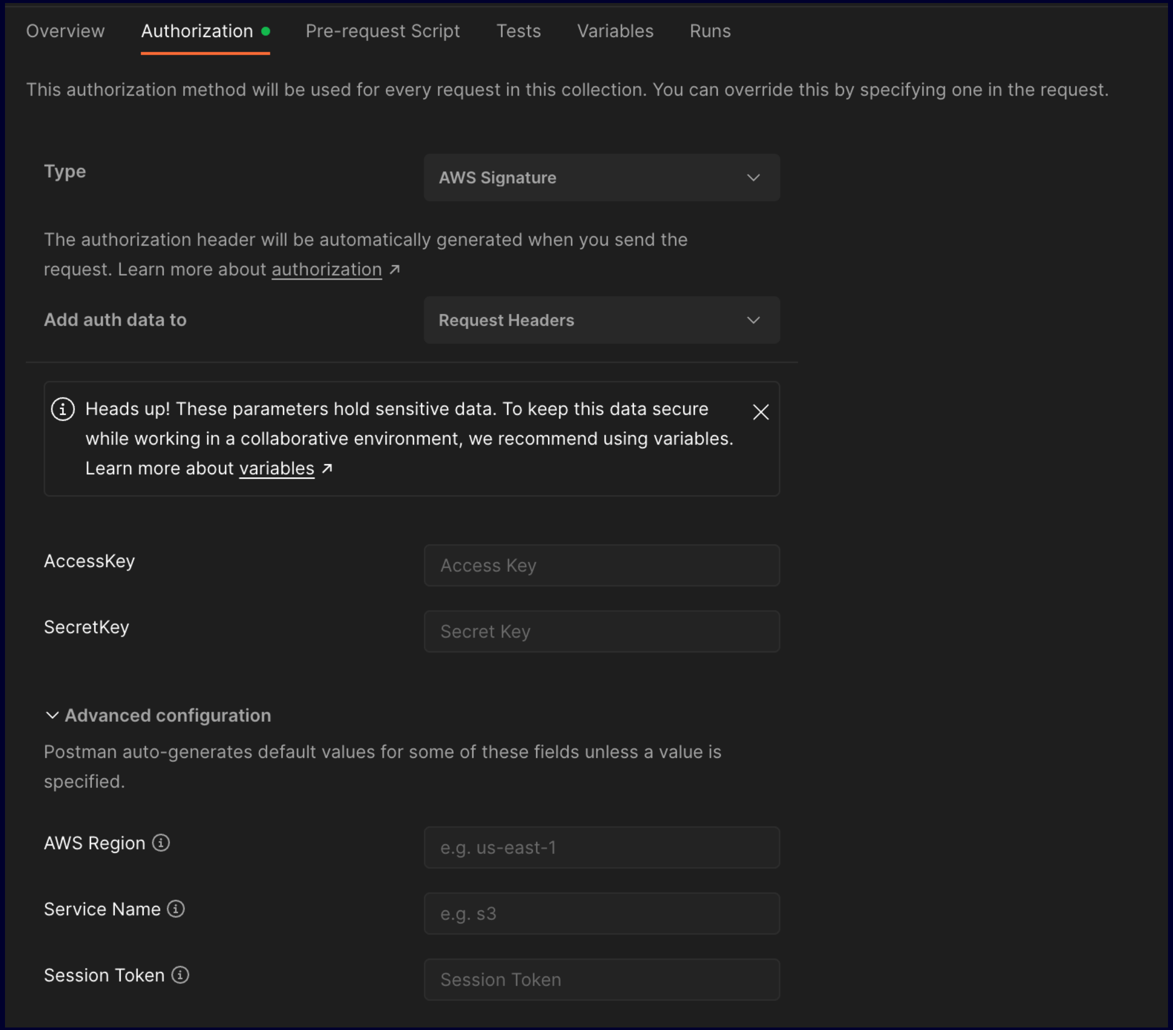


Demo: Cross Account



Sign Requests in Postman

Type:	AWS Signature
Add auth data to:	Header / Url
AccessKey:	<yourKey>
SecretKey:	<yourSecret>
AWS Region:	<ApiRegion>
Service Name:	execute-api
Session Token:	short term only



This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Type: AWS Signature

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

Add auth data to: Request Headers

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

AccessKey: Access Key

SecretKey: Secret Key

Advanced configuration

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

AWS Region: e.g. us-east-1

Service Name: e.g. s3

Session Token: Session Token

Summary

Offers strong security by using HMAC-SHA256 for signing insuring requests are authenticated, authorized, and tamper-proof

Signature validation is offloaded to AWS by piggybacking on IAM

By combining SigV4 with IAM policies, you can enforce detailed access permissions at the method and resource levels

No additional fees

Manual implementation without AWS SDKs is complex, especially in case of validating

API Gateway Resource Policies currently only supported by REST APIs

Performance Overhead: For large payloads, calculating the signature adds processing time

Additional resources

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-signing.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html

https://www.npmjs.com/package/aws4-axios

Thank you!

Sebastian Anton
santon@nepgroup.com

Icons used:

Infrastructure icons created by kerismaker - Flaticon
Microservices icons created by Uniconlabs - Flaticon