Unified authorization in microservices

(http and kafka-based interactions)

September 2021

STANISLAV DEVIATOV

Solution Architect

- 14 years experience in integration projects:
    - Mulesoft Anypoint Platform
    - Oracle Integration Stack (OSB, SOA/BPM Suites, ODI, OWB)
    - Open Source based solutions (Apache Camel, Apache Kafka)
- Wide experience: ERP and CRM implementation, ECM, IRM, MOLAP, Oracle database administration & development, java custom development, and microservice based projects.
- EPAM API & Integration CC expert
- Public Integration Community driver

Agenda

- Authentication and authorization
  - Basics
  - Sync & async interactions details
  - Typical cases
- HTTP security overview
- Kafka security overview
- Unified solution based on OAuth2.0 & OIDC

Assumptions

I assume that the audience is familiar with:

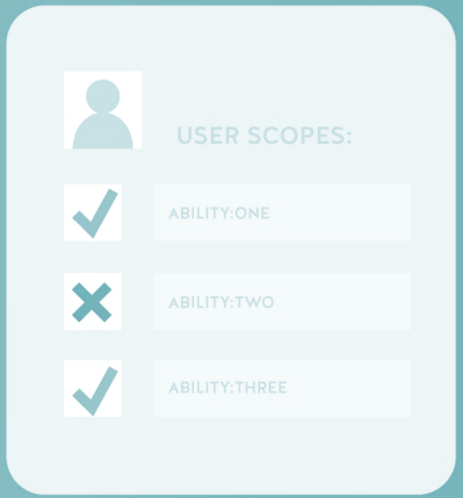
- HTTP
- Apache Kafka
- JWT
- OAuth2
    - OpenID Connect (OIDC)
    - User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization

AUTHENTICATION

WHO ARE YOU?

VS

AUTHORIZATION

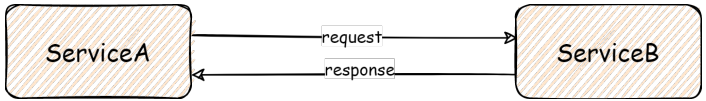


CAN YOU DO THAT?

Differences in authorization in sync and async communications

SYNC

- Direct communication:



- Simplified model
  - The callee gets authentication info from the caller

ASYNC

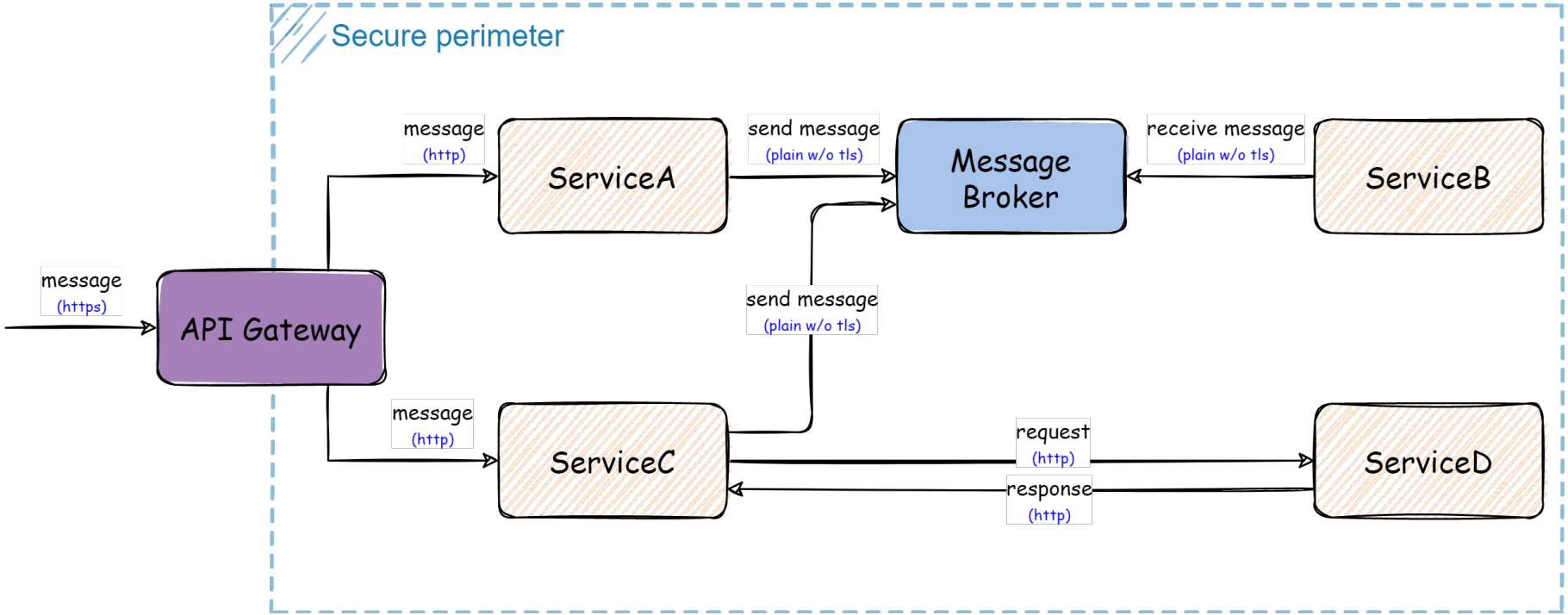
- Broker based communication:

- Simplified model
  - Producer has a write (or send) permission to the destination (queue/topic)
  - Consumer has a read (or receive) permission to the destination (queue/topic)
  - Consumer doesn’t get authentication info from the producer

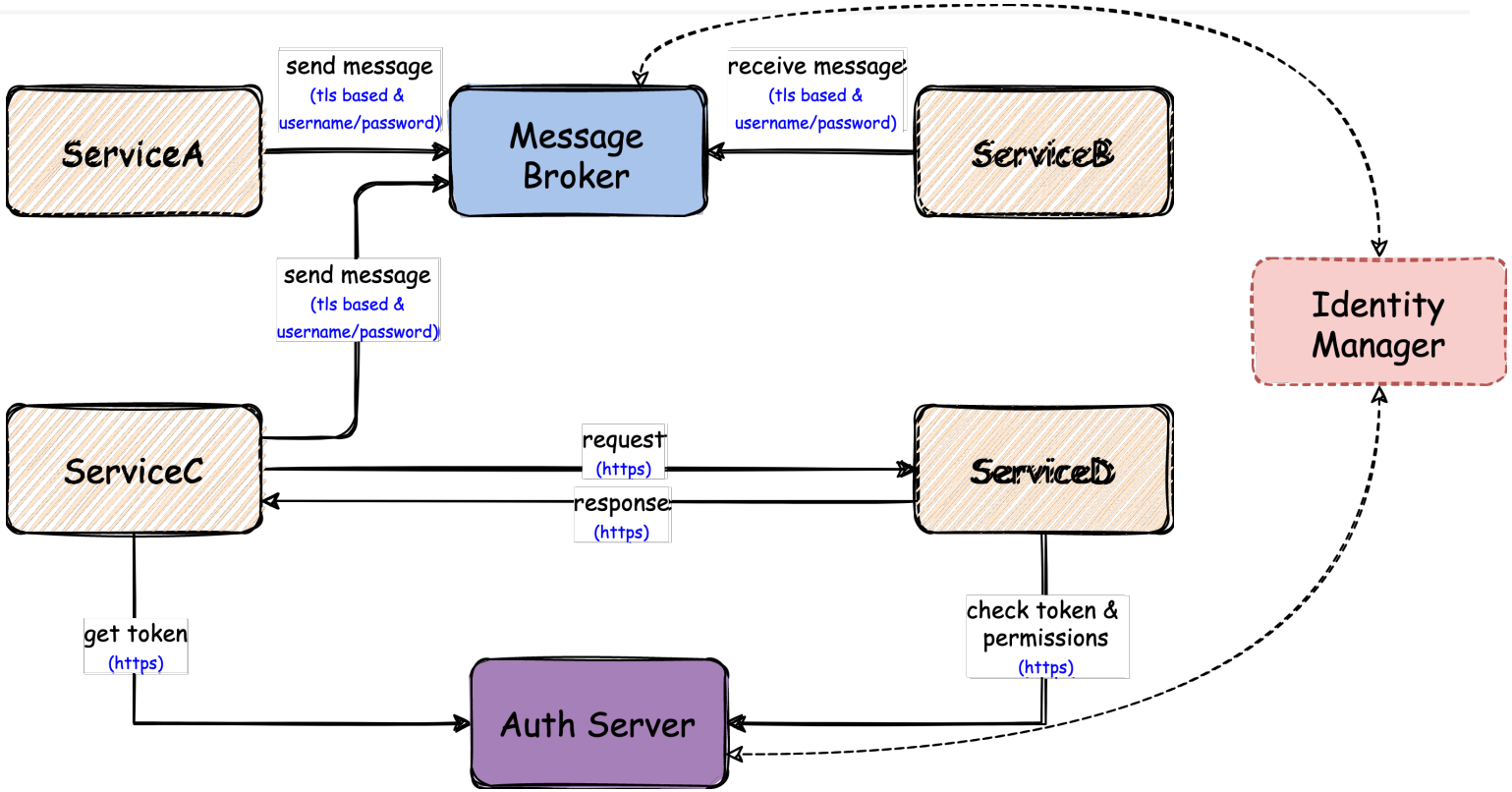
TYPICAL AAA* IN MICROSERVICE BASED PROJECTS

** authentication, authorization, and accounting*

Option 1: perimeter security only



Option 2: transport and application-level security



Option 3: your case

HTTP AND KAFKA SECURITY

Security Tokens

JWT

- JSON Web Tokens (pronounced /dʒɔt/, same as the word "jot")
- RFC-7523 published in 2014
- Based on JSON
- JWT, OAuth2, and OpenID Connect are common in the Social Media, but is now finding its way into the enterprise
- It's used in:
    - OpenID Connect (OIDC)
    - OAuth 2.0

SAML

- Security Assertion Markup Language Tokens
- SAML 2 introduced in 2005
- Based on XML
- SAML 2 is the standard for Enterprise SSO
- It's used commonly in:
    - WS-Security
    - WS-Trust
    - WS-Federation
    - Etc.
- It uses:
    - XML Digital Signature
    - XML Encryption
    - XML Schema (XSD)

HTTP Security

TRANSPORT LEVEL

- TLS based communications

APPLICATION LEVEL

- HTTP Strict Transport Security (HSTS)
- Security Headers
- Cross-origin resource sharing (CORS)
- Security Tokens
- API Keys
- Input validation
- etc.

Kafka Security – Authentication & authorization

AUTHENTICATION

- Authentication of connections to brokers from clients (producers and consumers), other brokers and tools, using either SSL or SASL*. Kafka supports the following SASL mechanisms:
  - SASL/GSSAPI (Kerberos) - starting at version 0.9.0.0
  - SASL/PLAIN - starting at version 0.10.0.0
  - SASL/SCRAM-SHA-256 and SASL/SCRAM-SHA-512 - starting at version 0.10.2.0
  - SASL/OAUTHBEARER - starting at version 2.0

AUTHORIZATION

- Kafka ships with a pluggable Authorizer and an out-of-box authorizer implementation that uses zookeeper to store all the ACLs.

** Simple Authentication and Security Layer (SASL) is defined by RFC 4422*

Kafka Security Model

TOPIC

- Write
- Read
- Describe
- Create
- Delete
- DescribeConfigs
- AlterConfigs
- IdempotentWrite

GROUP

- Read
- Describe
- Delete

CLUSTER

- Create
- Describe
- Alter
- DescribeConfigs
- AlterConfigs
- IdempotentWrite
- ClusterAction

TRANSACTIONAL ID

- Describe
- Write

DELEGATION TOKEN

- Describe

Kafka Security - SASL/OAUTHBEARER

KIP-255

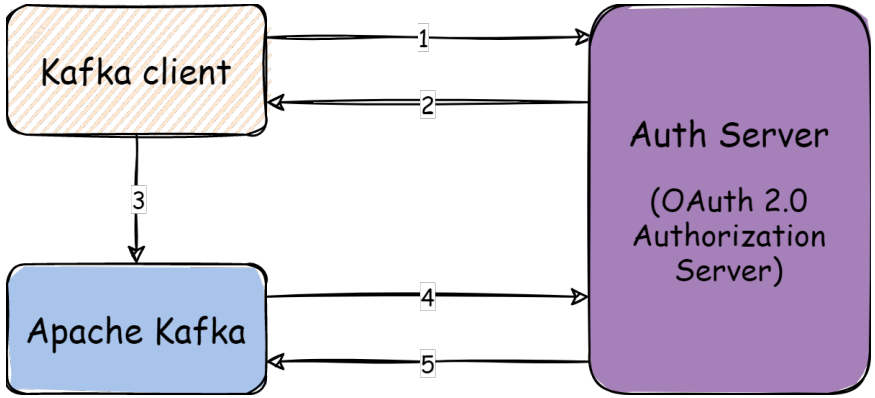
- KIP-255 OAuth Authentication via SASL/OAUTHBEARER
- Adopted in 2.0

KIP-368

- KIP-368: Allow SASL Connections to Periodically Re-Authenticate
- Adopted in 2.2
- Kafka connections are long-lived, so there is no ability to change the bearer token associated with a particular connection. Allowing SASL connections to periodically re-authenticate would resolve this.
- Additional benefits:
  - eliminate access to Kafka for connected clients, the current requirement is to remove all authorizations
  - the use of short-lived tokens is a common OAuth security recommendation

Kafka Authorizer for OAuth2

- Strimzi Kafka OAuth
  - Details:
    - https://github.com/strimzi/strimzi-kafka-oauth
  - Provides server-side and client-side parts:
    - custom server callback that will provide server-side token validation
    - custom client callback that will automatically perform re-authentication if access token expires



SASL/OAUTHBEARER – Client library support

JAVA

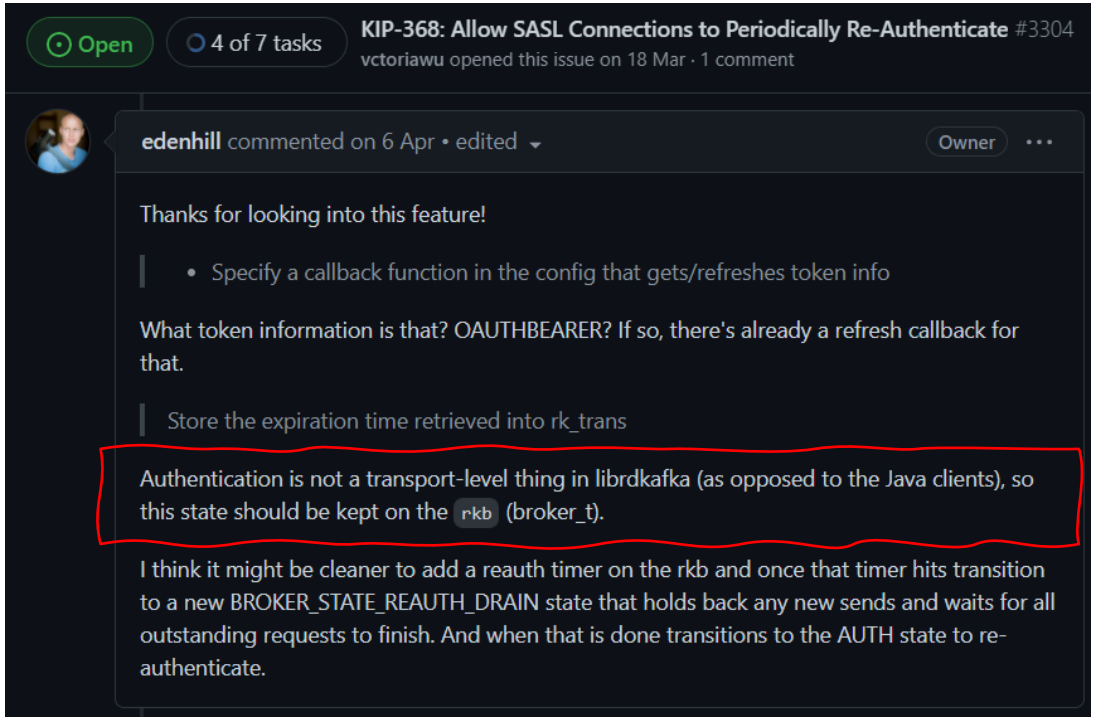
- Apache Kafka client
- Strimzi client lib (callback handler)

NON-JAVA

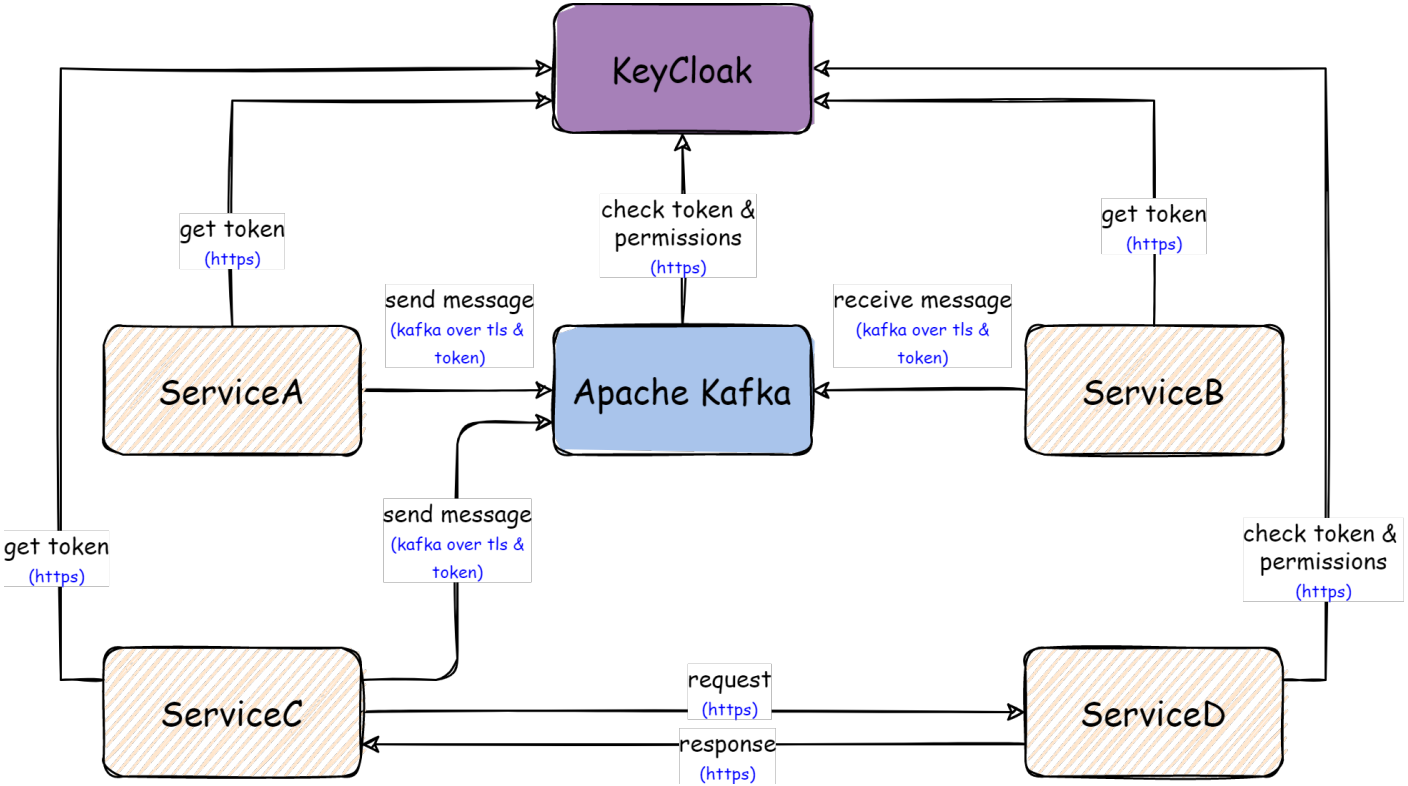
- librdkafka
  - Is the Apache Kafka C/C++ client library
  - *Supports KIP-255 only*
  - Has language bindings
    - C#/.NET: confluent-kafka-dotnet (based on rdkafka-dotnet)
    - Go: confluent-kafka-go
    - Lua: luardkafka
    - Node.js: node-rdkafka
    - Python: confluent-kafka-python
    - Python: PyKafka
    - Rust: rust-rdkafka
    - Shell: kafkacat - Apache Kafka command line tool
    - Etc. https://github.com/edenhill/librdkafka#language-bindings

SASL/OAUTHBEARER - KIP-368 in librdkafka

KIP-368 issue: https://github.com/edenhill/librdkafka/issues/3304



Unified solution



DEEP DIVE INTO
KAFKA SASL/OAUTHBEARER

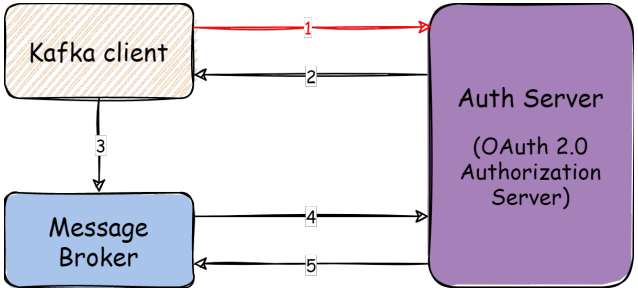
How it works? #1

POST http://keycloak:8080/auth/realms/kafka-authz/protocol/openid-connect/token

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

	KEY	VALUE
☑	grant_type	client_credentials
☑	client_id	team-b-client
☑	client_secret	team-b-client-secret
	Key	Value

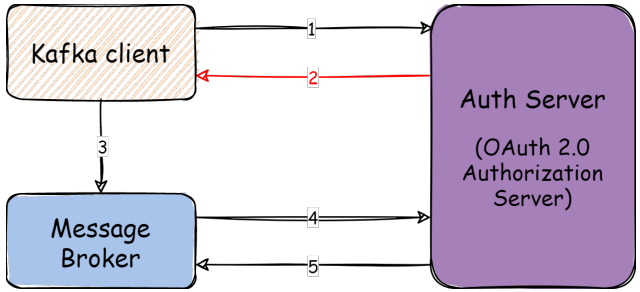


How it works? #2

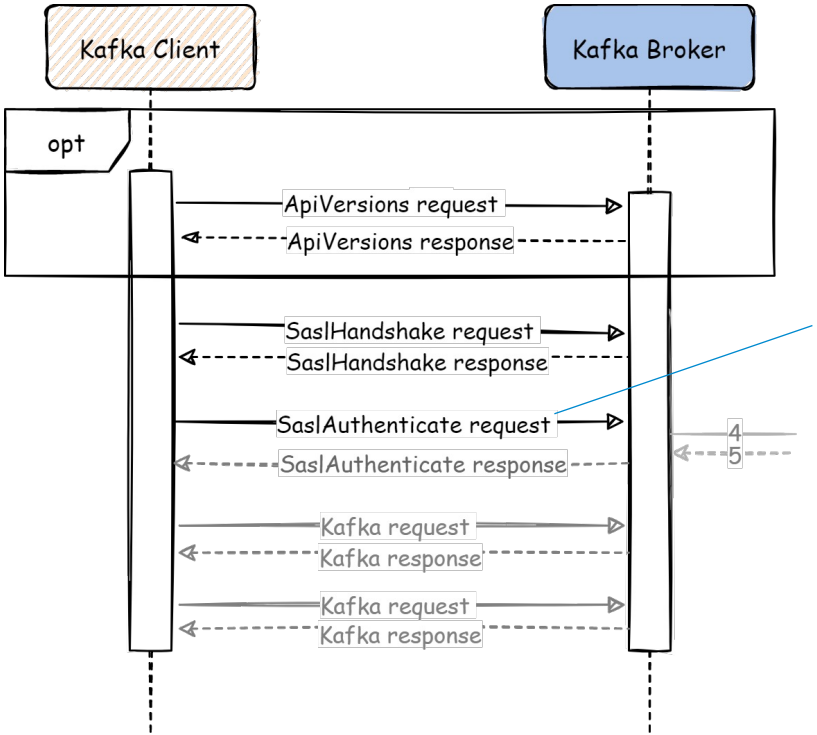
Body Cookies Headers (12) Test Results — Status: 200 OK Time: 45 ms Size: 1.8 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJZbkUtRXlYSzVHTVEycVRyb240WFRVY1pNUXdhcXVjZGRGd3JpM1hDSHhrIn0.
        eyJleHAiOjE2MjU0MDExNDMsImlhdCI6MTYyNTQwMTAyMywianRpIjoiMDM2N2JjMWQtYjk0ZC00ODg2LTljMTgtNGY0OTkzMGEyM2I5IiwiaXNzIjoiaHR0cDovL2tleWNsb2FrOjgwODAvYXV0aC9yZWFsbXMva2Fma2EtYXV0aHoiLCJhdWQiOiJhY2NvdW50Iiwic3ViIjoiNjM1MTRhY2YtYzUwZi00NDM0LTllZTMtYWViZDEwODU2N2VlIiwidHlwIjoiQmVhcmVyIiwiYXpwIjoidGVhbS1iLWNsaWVudCIsImFjciI6IjEiLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsib2ZmbGluZV9hY2Nlc3MiLCJEZXYgVGVhbSBCIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJlbWFpbCBwcm9maWxlIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJjbGllbnRIb3N0IjoiMTcyLjE4LjAuMSIsImNsaWVudElkIjoidGVhbS1iLWNsaWVudCIsInByZWZlcnJlZF91c2VybmFtZSI6InNlcnZpY2UtYWNjb3VudC10ZWFtLWItY2xpZW50IiwiY2xpZW50QWRkcmVzcyI6IjE3Mi4xOC4wLjEifQ.
        f50qP9FwMWJoYZhaKaJDvJq4f6s9KeVRShpIfJMuC2hehX0ohVMAME9pnUcbH1Yr7HcrtZgSDJxHQ0iUh9pZ_XvPbiJzYl8sk5GqARGaOnGbHf7jcU_fAlHUY5LyAm8I1wxlF7m1CHDTSzA5jknfLsvxlG-9sfK__s7wXxovyyHqXFUMuMtUdHbKagZkzfK1MZXJTujtxzBvKOF_yLfTfWypTxKwv3X-bKTu4cm_ax20xQMuKH_B41pgjjpys9eUfko75MOV
        3RQ8GmbsNftTZVl6n8K6zPiJ8sH5YrGlrTs2KFC_-crSJMSe4-6Bmme22_nTyP98JQsBodQc9RrKTg",
    "expires_in": 120,
    "refresh_expires_in": 0,
    "token_type": "Bearer",
    "not-before-policy": 0,
    "scope": "email profile"
}
```

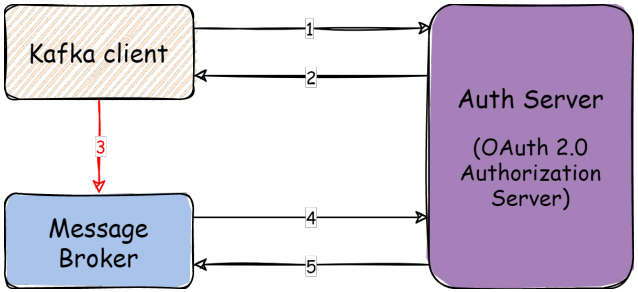
How it works? #3



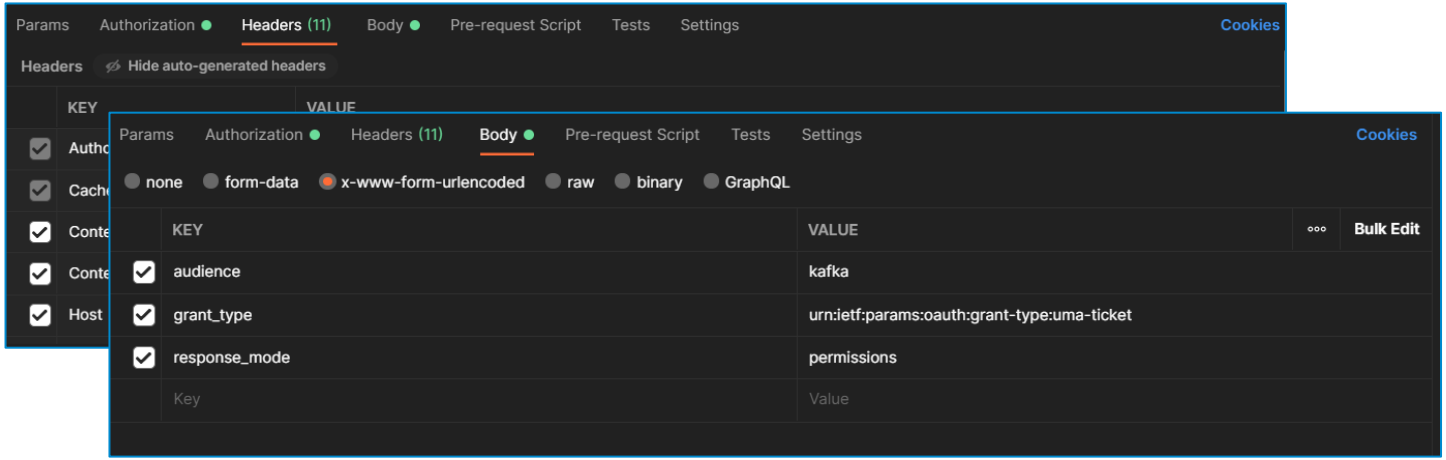
including assess token from #2

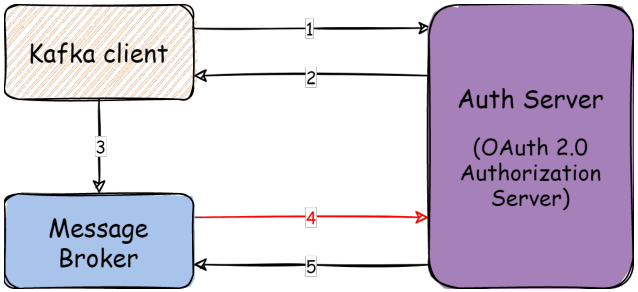
More details:

- https://kafka.apache.org/protocol#sasl_handshake
- https://www.novatec-gmbh.de/en/blog/kafka-security-behind-the-scenes/
- https://datatracker.ietf.org/doc/html/rfc4422



How it works? #4





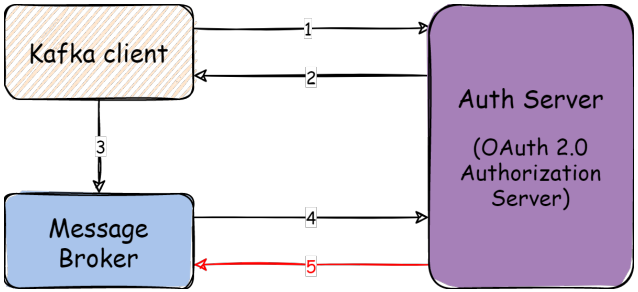
How it works? #5

Params | Authorization | Headers (11) | Body | Pre-request Script | Tests | Settings | Cookies

Body | Cookies | Headers (11) | Test Results | Status: 200 OK | Time: 107 ms | Size: 883 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
        ],
        "rsid": "ca24cc58-47e7-4ac0-a843-6954f9063d7f",
        "rsname": "kafka-cluster:cluster2,Group:*"
    },
    {
        "scopes": [
            "ClusterAction",
            "DescribeConfigs",
            "AlterConfigs"
        ],
        "rsid": "190e4134-bd7b-4928-9b5e-9da847fff8a7",
        "rsname": "kafka-cluster:cluster2,Cluster:*"
    },
    {
        "scopes": [
            "Alter",
            "Delete",
            "Describe",
            "Write",
            "Read",
            "IdempotentWrite",
            "Create",
            "DescribeConfigs",
            "AlterConfigs"
        ],
        "rsid": "f98d2535-bb30-418f-a51e-5292732b1e73",
        "rsname": "kafka-cluster:cluster2,Topic:*"
    }
]
```

HOW COULD IT BE CONSISTENT FOR HTTP?

HTTP interceptor to check permissions #1

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

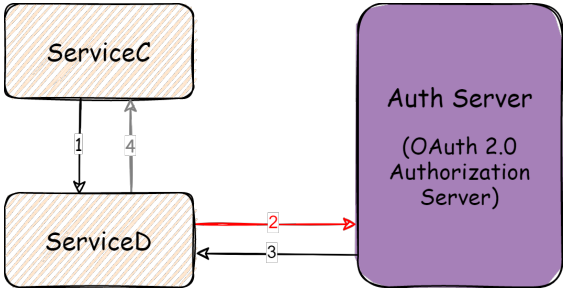
Headers Hide auto-generated headers

KEY	VALUE	Bulk Edit	Presets
Authorization	Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJLY1JWY1gxZHVndVc0bnBYT3VQRzg1emdaeHVremx6Z2VSTDg3SHdvVWEwIn0.e...		
Cache-C			
Content-			
Content-			
Host			

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	Bulk Edit
audience	service-d	
grant_type	urn:ietf:params:oauth:grant-type:uma-ticket	
response_mode	permissions	
Key	Value	

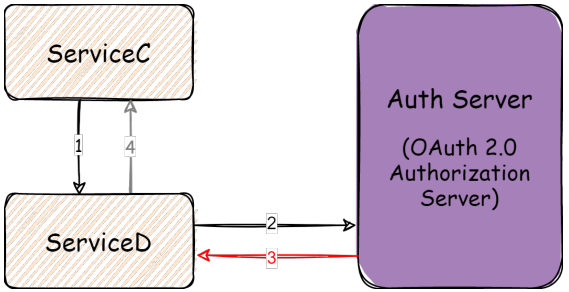


HTTP interceptor to check permissions #2

Body Cookies Headers (11) Test Results 200 OK 18 ms 657 B

Pretty Raw Preview Visualize JSON


```
[
    {
        "scopes": [
            "POST",
            "GET"
        ],
        "rsid": "901de5d1-de6f-40f5-b46e-26b11d3faa65",
        "rsname": "/v1/invoices"
    },
    {
        "scopes": [
            "DELETE",
            "GET",
            "PUT"
        ],
        "rsid": "24e3beea-13fc-4aea-aa19-cc336e6a06bf",
        "rsname": "/v1/invoices/{id}"
    },
    {
        "scopes": [
            "GET"
        ],
        "rsid": "2c43f1d4-e6a2-46b9-9d25-6772f194c096",
        "rsname": "/v1/users"
    }
]
```

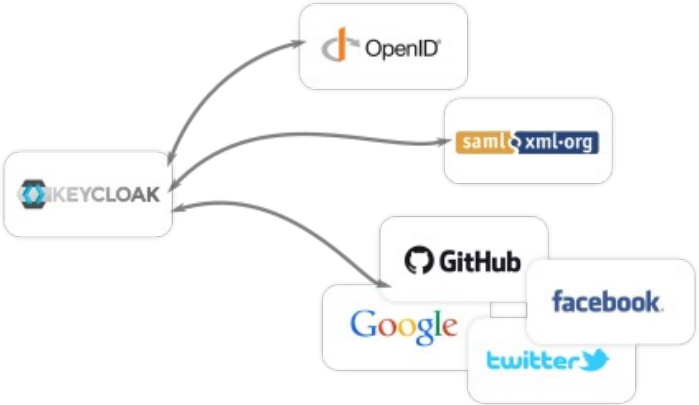



AUTHORIZATION SERVER (KEYCLOAK)

KeyCloak

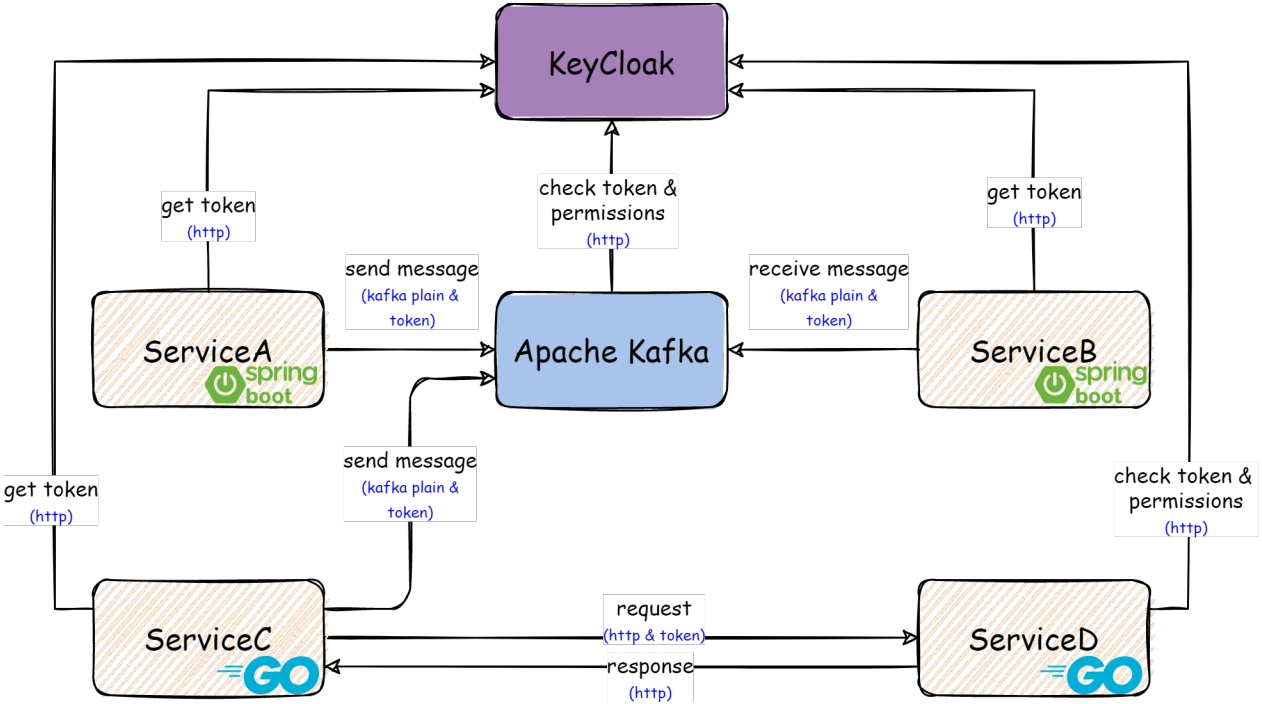
- Supports protocols:
  - OpenID Connect 1.0
  - OAuth 2.0
  - SAML 2.0
- Supports access control mechanisms:
  - Attribute-based access control (ABAC)
  - Role-based access control (RBAC)
  - User-based access control (UBAC)
  - Context-based access control (CBAC)
  - Rule-based access control using JavaScript
  - Time-based access control
- Red Hat Single Sign-On (RH-SSO) is RedHat supported version of KeyCloak

- Single Sign-On (SSO):
  - Single Log Out (SLO)
  - Kerberos bridge
- User Federation:
  - Built-in support to connect to existing LDAP or Active Directory servers
- Identity Brokering and Social Login



DEMO

Demo case



Sources: https://github.com/stn1slv/meetup-authorization

Possible disadvantages

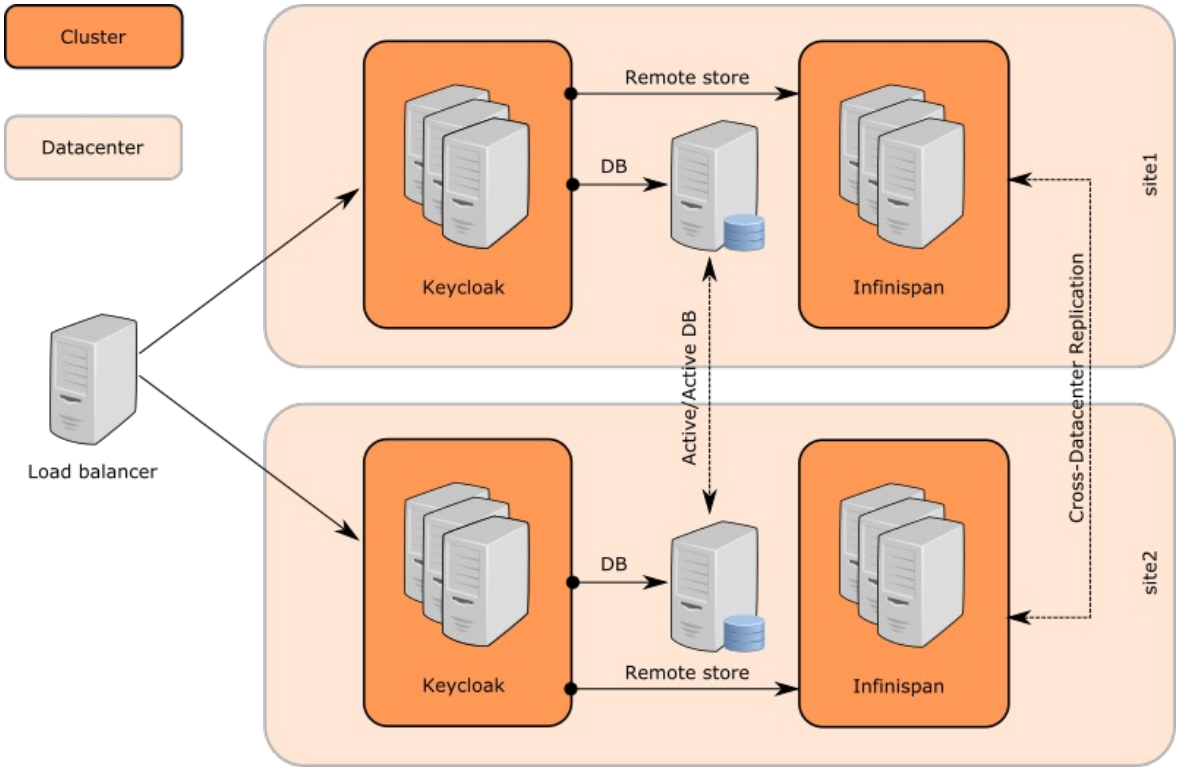
- KIP-368: Allow SASL Connections to Periodically Re-Authenticate support is limited for non-JVM platforms
- Infrastructure components don't support SASL/OAUTHBEARER and should be improved. For example:
  - danielqsj/kafka-exporter
  - lightbend/kafka-lag-exporter
  - Confluent Schema Registry (w/o commercial components)
  - Confluent REST proxy (w/o commercial components)
- Depends on authorization service
  - Interaction with Auth service for each HTTP call
  - Interaction with Auth service for each Kafka client authentication (and re-authentication)

THANK YOU

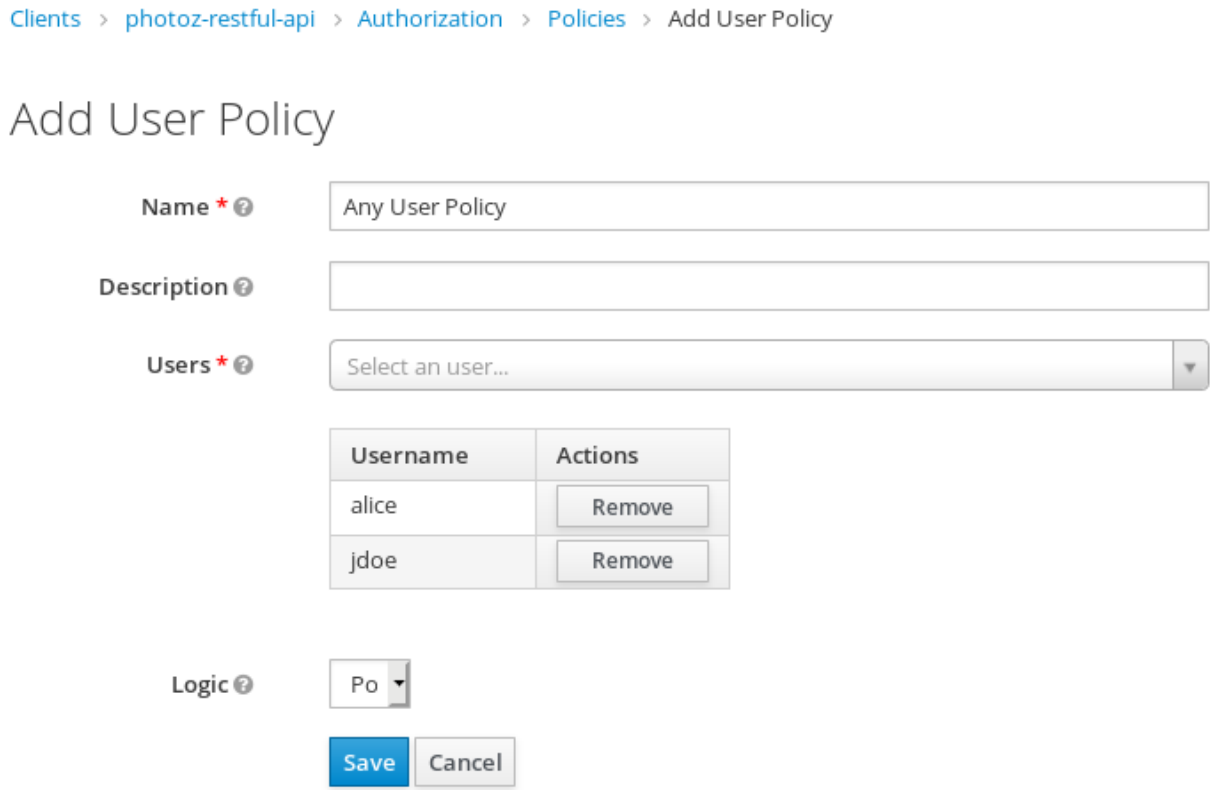
Q&A

APPENDIX

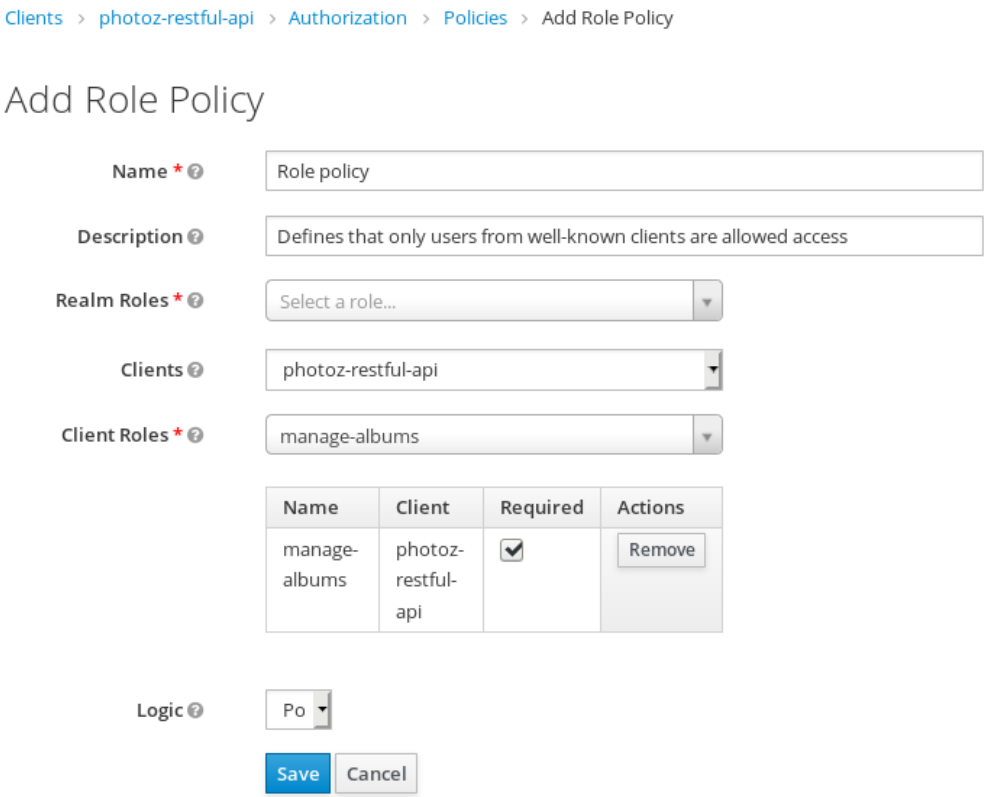
KeyCloak clustering



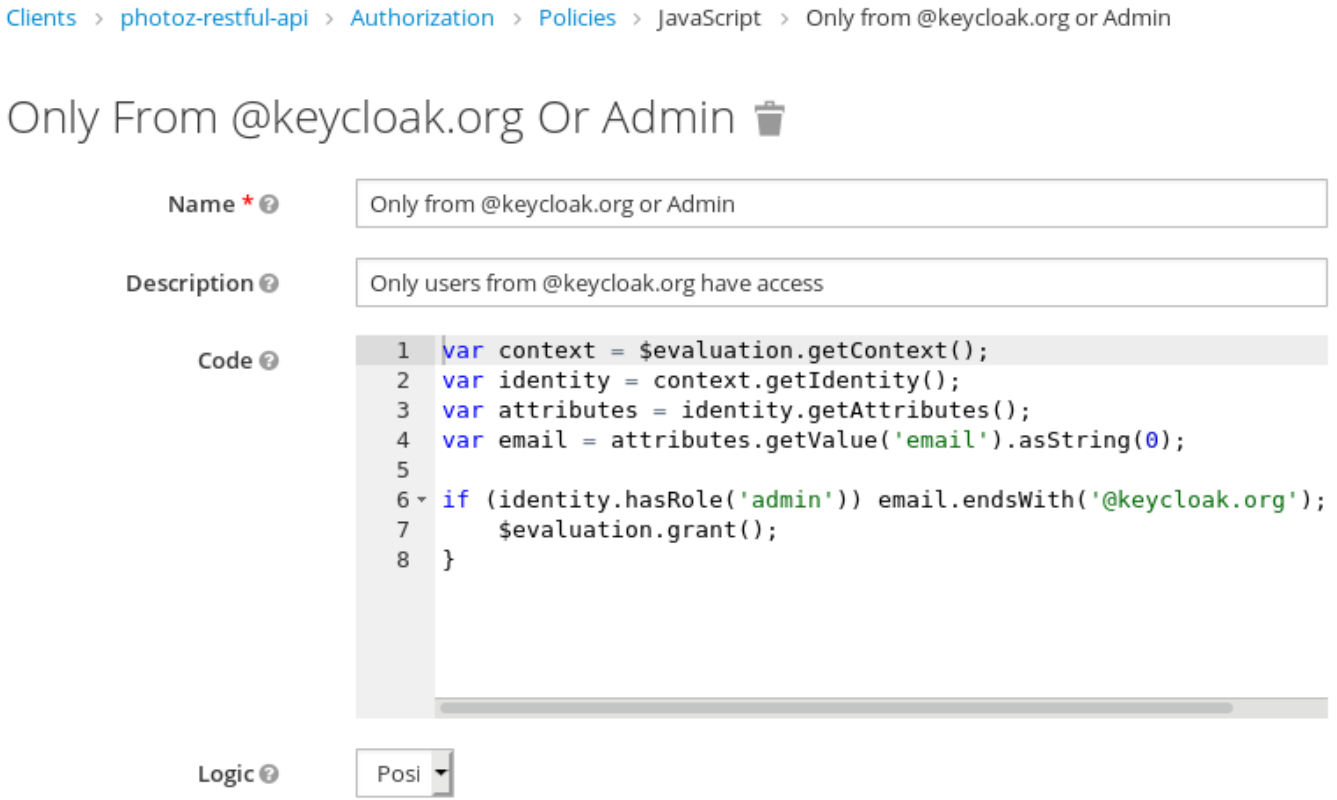
KeyCloak: User-Based Policy



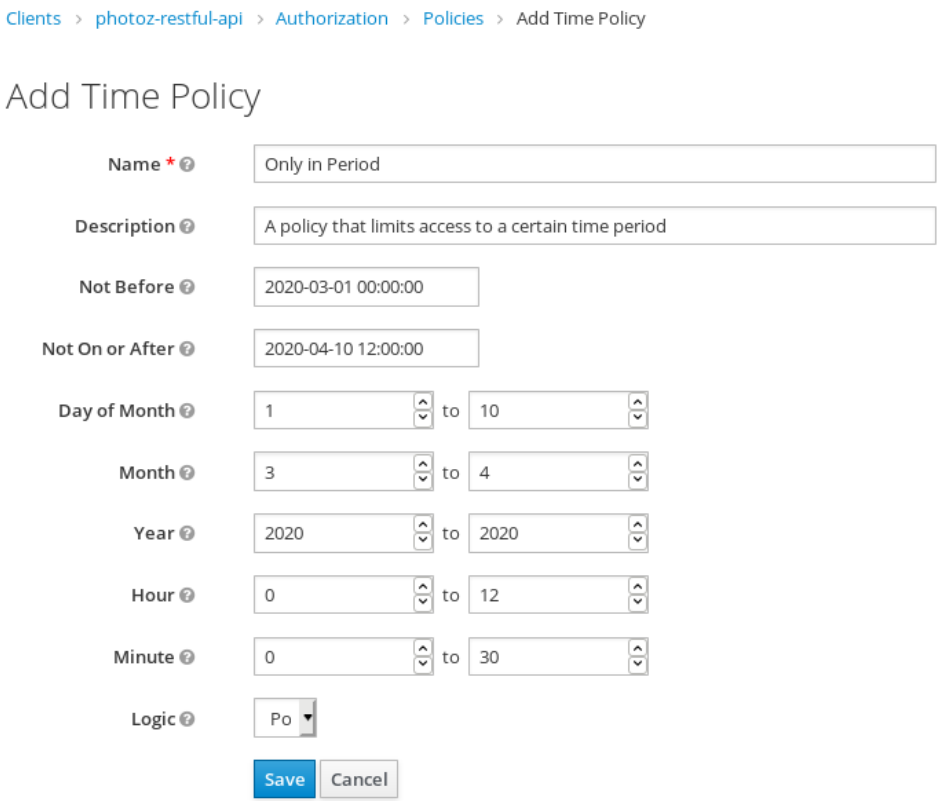
KeyCloak: Role-Based Policy



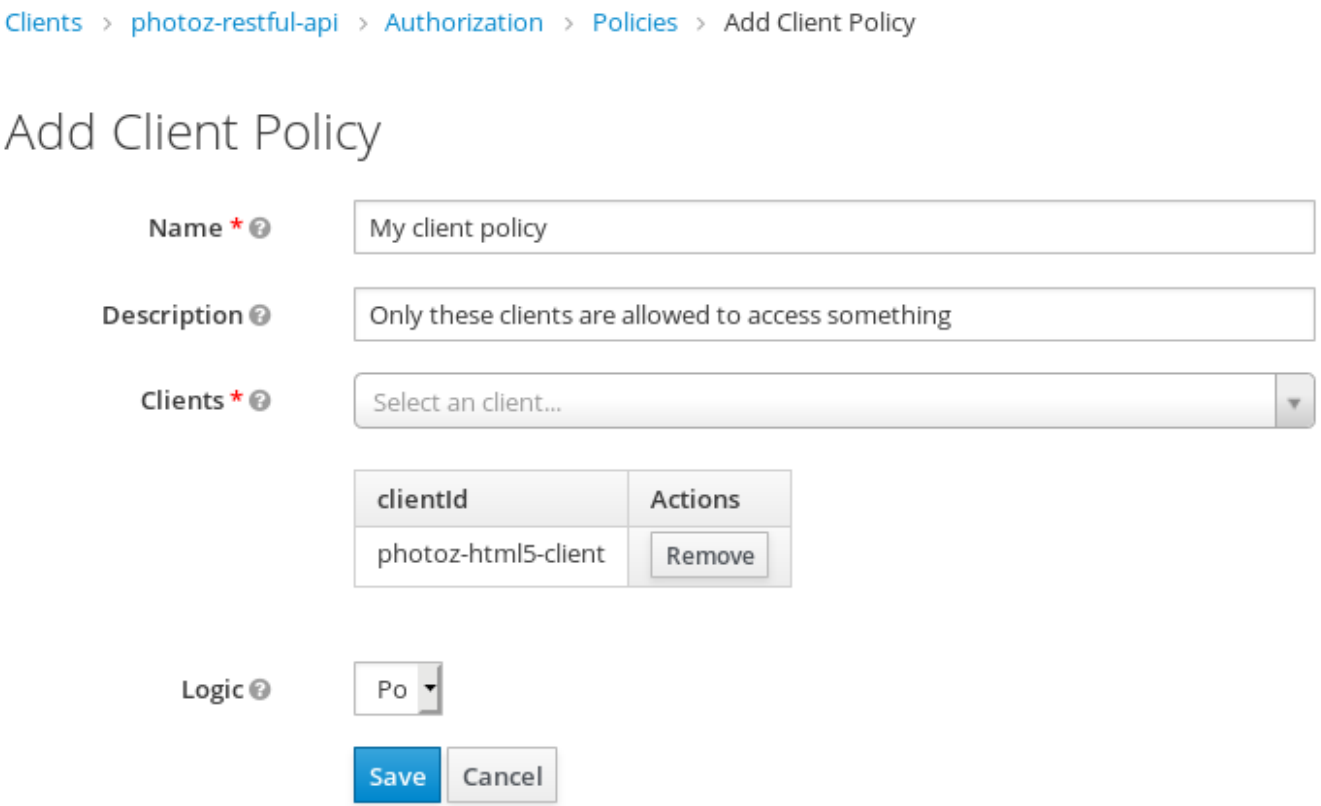
KeyCloak: JavaScript-Based Policy



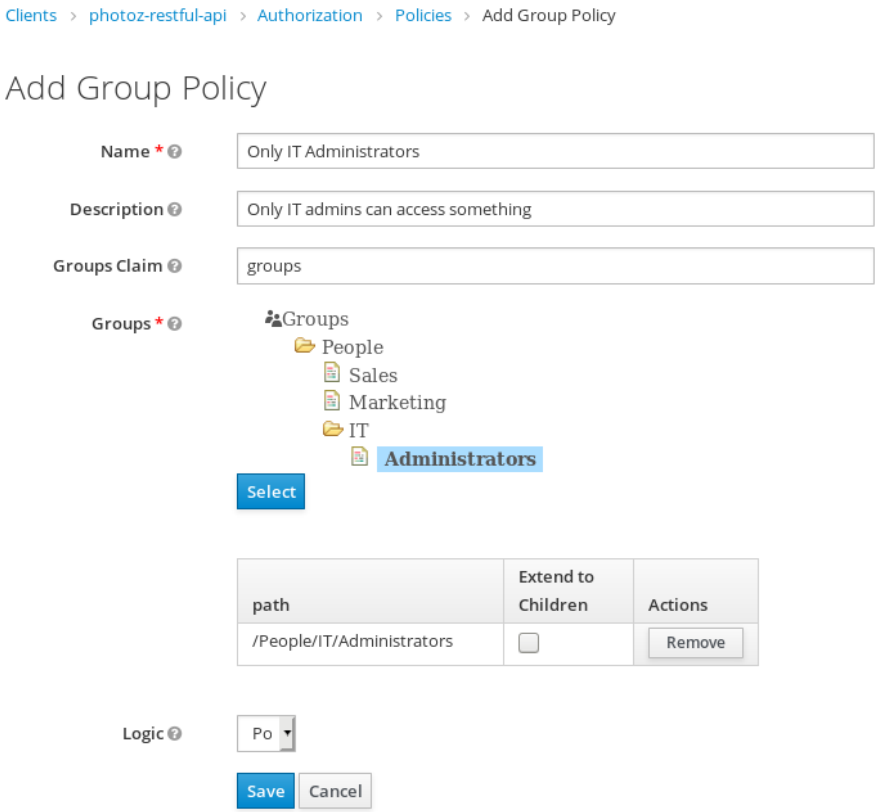
KeyCloak: Time-Based Policy



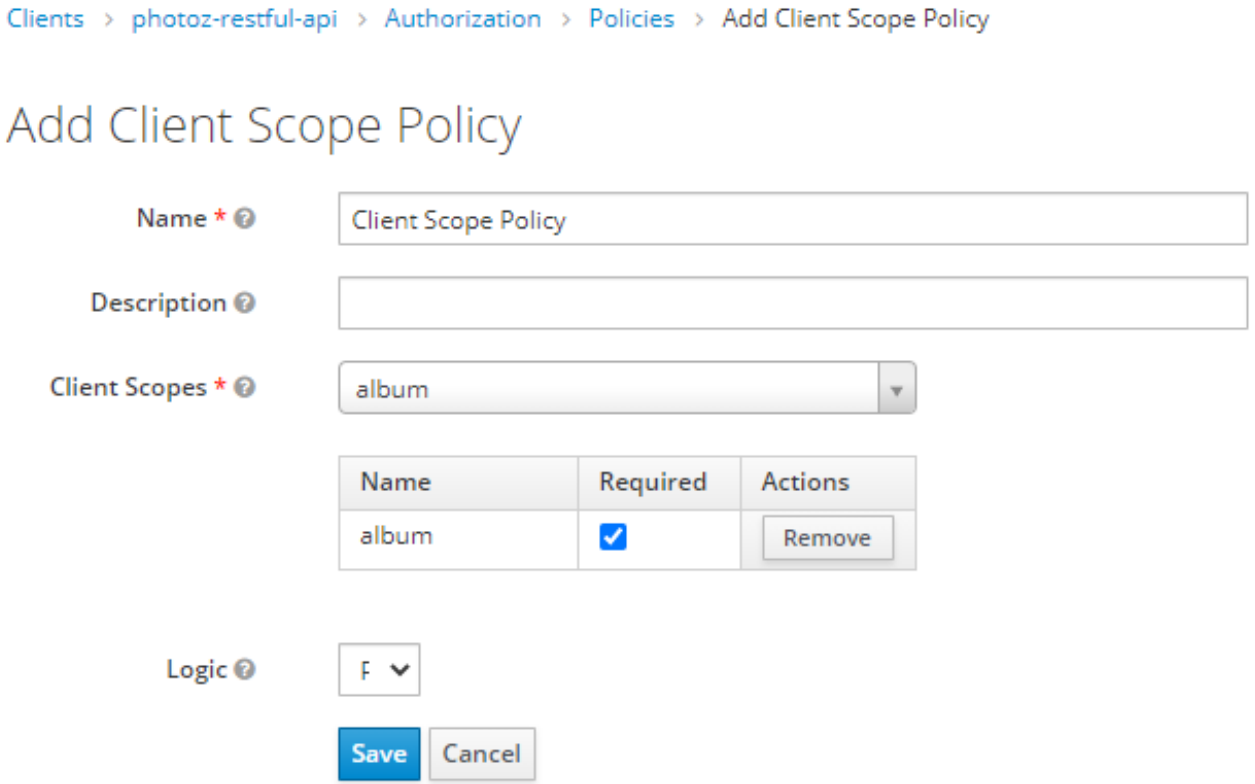
KeyCloak: Client-Based Policy



KeyCloak: Group-Based Policy



KeyCloak: Client Scope-Based Policy



KeyCloak: Aggregated Policy