KONG AND KEYCLOAK

Securing API through OIDC

INTRODUCTION

We will see how to configure a Kong API Gateway with the OIDC (OpenID Connect) Plugin and Keycloak to secure APIs.

We will be using docker-compose to build the infrastructure to implement this.

We have

- Kong - An API Gateway
- Konga – A GUI to Kong Admin API
- Keycloak - A OpenID Connect Provider (OP)

But first let’s see what is oidc?

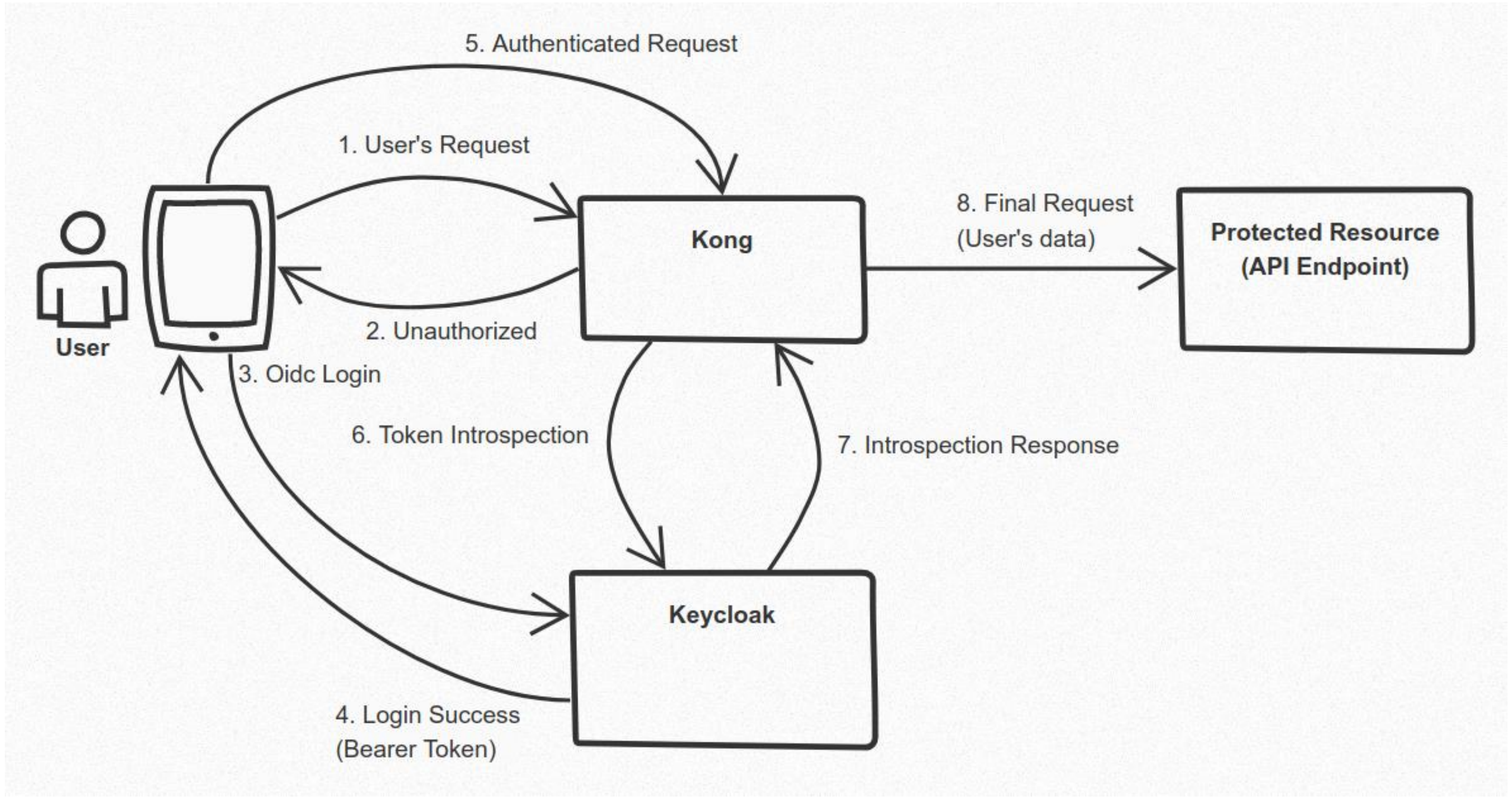
OIDC

OpenID is a simple level of identity implemented above the OAuth 2.0 protocol: it allows its Clients to verify the identity of the end user, based on the authentication performed by an Authorization Server, as well as to obtain basic information on the user profile.

With a Security Token Service (STS), the RP is redirected to an STS, which authenticates the RP and issues a security token that grants access, instead of the application that directly authenticates the RP. Claims are extracted from tokens and used for identity-related activities.

The OpenID standard defines a situation in which a cooperating site can act as an RP, allowing the user to access multiple sites using a set of credentials. The user benefits from not having to share access credentials with multiple sites and the operators of the collaborating site must not develop their own access mechanism.

THE INFRASTRUCTURE



KONG OIDC

First, we need to create a Kong image with the kong-oidc plugin installed. We will do this by creating a Dockerfile and building the image from that.

Dockerfile:

```
FROM kong:2.0.0-alpine

LABEL description="Alpine + Kong 2.0.0 + kong-oidc plugin"

USER root

RUN apk update

RUN apk add build-base bash curl unzip git

RUN apk add lua5.1 lua5.1-dev
RUN apk add openssl openssl-dev

# Build Luarocks.
RUN cd /tmp && \
    git clone https://github.com/keplerproject/luarocks.git && \
    cd luarocks && \
    sh ./configure && \
    make build install && \
    cd && \
    rm -rf /tmp/luarocks

RUN luarocks install kong-oidc

USER kong
```

```
docker build -t kong:2.0.0-alpine-oidc .
```

BUILDING THE INFRASTRUCTURE 1/3

We will build each service/container from the shown docker-compose.yml file.

We will not focus on what’s in the docker-compose.yml file, butfocus more on how to build the infrastructure from it and the keycloak part since we assume you know how to setup kong.

First we start the kong-db service (the database server Kong will use):

```
docker-compose up –d kong-db
```

Then we launch the kong migrations:

```
docker-compose run --rm kong kong migrations bootstrap
```

And finally we can start kong(with the oidc plugin installed)

```
docker-compose up -d kong
```

BUILDING THE INFRASTRUCTURE 2/3

Next, we start Konga(covered in the previous presentation) which will be listening on port 1337 (http://localhost:1337)

```
docker-compose up -d konga
```

We assume that you know how to setup a service and a route for Kong, so we will skip that part and note that to test the system, we will use Mockbin (a service that generates endpoints to test HTTP requests, responses, sockets and APIs).

BUILDING THE INFRASTRUCTURE 3/3

Finally we start Keycloak:

We start the keycloak database service:

```
docker-compose up -d keycloak-db
```

We start the keycloak service:

```
docker-compose up -d keycloak
```

We check that everything is standing with:

```
docker-compose ps
```

We should see all the containers running (state should be up).

Keycloak will be available at the url http://localhost:8180.

SETTING UP KEYCLOAK

We will show the following:

- Add a Realm
- Add a Client
- Add a User

SETUP KONG TO WORK WITH KEYCLOAK

We will do the following:

- Configure the OIDC plugin on Kong
- Test the OIDC functionality with Kong as a client of Keycloak

THE END

And that's it! Now let's test it with the web browser. When you try to hit http://localhost:8000/test, now, Kong should redirect you to log in to Keycloak. Log in as the user you created in Keycloak earlier, and you should be redirected to the original page you requested.