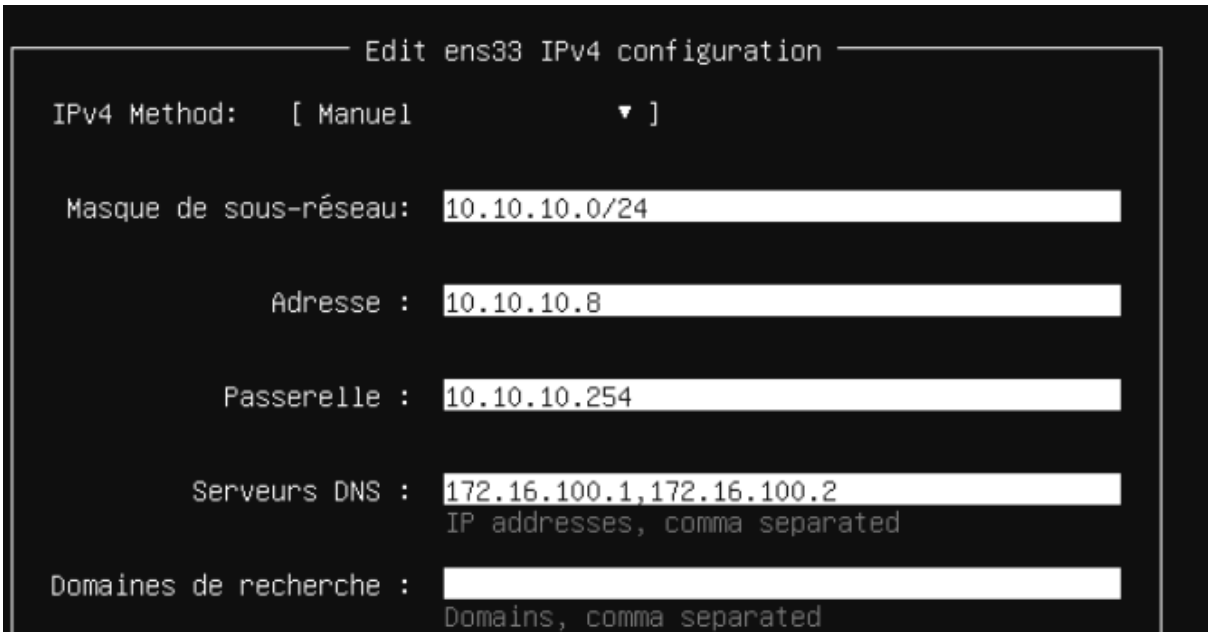
Authentication Guacamole-VPN

Solution recommandée : **Utiliser OpenLDAP avec Guacamole**



Methode 1 créer guacamole extensions : pas marcher car c’est plus poussé et besoin d'un développeur

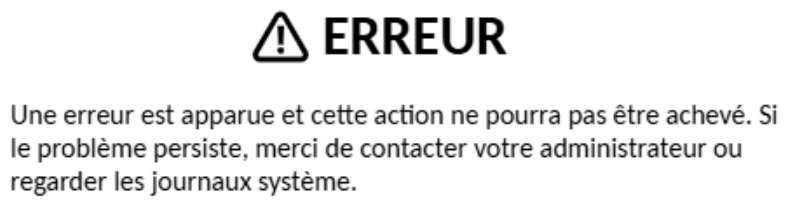
```
afar@apache-guaca:~# ls
pacheguac.conf  apacheguac.csr  dead.letter
pacheguac.crt   apacheguac.key  guacamole-auth-extension
afar@apache-guaca:~# cd guacamole-auth-extension/
afar@apache-guaca:~/guacamole-auth-extension# ls
om.xml  src
afar@apache-guaca:~/guacamole-auth-extension# cd src/
afar@apache-guaca:~/guacamole-auth-extension/src# ls
ain
afar@apache-guaca:~/guacamole-auth-extension/src# cd main/
afar@apache-guaca:~/guacamole-auth-extension/src/main# ls
ava  resources
afar@apache-guaca:~/guacamole-auth-extension/src/main# cd resources/
afar@apache-guaca:~/guacamole-auth-extension/src/main/resources# ls
afar@apache-guaca:~/guacamole-auth-extension/src/main/resources# nano guac-manifest.json
afar@apache-guaca:~/guacamole-auth-extension/src/main/resources# cd
afar@apache-guaca:~# cd guacamole-auth-extension/
afar@apache-guaca:~/guacamole-auth-extension# ls
om.xml  src
afar@apache-guaca:~/guacamole-auth-extension# mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -------------< org.apache.guacamole:guacamole-auth-custom >-------------
[INFO] Building guacamole-auth-custom 1.5.5
[INFO] --------------------------------[ jar ]---------------------------------
ownloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resource
-plugin/2.6/maven-resources-plugin-2.6.pom
ownloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-resources
```

```
ng-2.1.jar (208 kB at 3.6 MB/s)
[INFO] Building jar: /home/zafar/guacamole-auth-extension/target/guacamole-auth-custom-1.5.5.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.061 s
[INFO] Finished at: 2025-01-29T08:12:39Z
[INFO] ------------------------------------------------------------------------
zafar@apache-guaca:~/guacamole-auth-extension# cp target/guacamole-auth-custom-1.5.5.jar /etc/guacamol
```

```
zafar@apache-guaca:~/guacamole-auth-extension# tree
.
├── pom.xml
├── src
│   └── main
│       ├── java
│       │   └── org
│       │       └── apache
│       │           └── guacamole
│       │               └── auth
│       │                   └── TutorialAuthenticationProvider.java
│       └── resources
│           └── guac-manifest.json
└── target
    ├── classes
    │   ├── guac-manifest.json
    │   └── org
    │       └── apache
    │           └── guacamole
    │               └── auth
    │                   └── TutorialAuthenticationProvider.class
    ├── generated-sources
    │   └── annotations
    ├── guacamole-auth-custom-1.5.5.jar
    ├── maven-archiver
    │   └── pom.properties
    └── maven-status
        └── maven-compiler-plugin
            └── compile
                └── default-compile
                    ├── createdFiles.lst
                    └── inputFiles.lst

21 directories, 9 files
```

Sources

[Custom authentication — Apache Guacamole Manual v1.5.5]



Cette méthode ne répond pas à ma demande d'authentification, car elle est trop complexe. En effet, il est difficile pour les équipes informatiques de gérer manuellement les utilisateurs en modifiant les fichiers à chaque ajout ou modification. Créer chaque utilisateur à la main dans les fichiers de configuration devient rapidement ingérable et peu évolutif, surtout quand le nombre d'utilisateurs augmente.

Cette rechache et mis en place m’a appris de comment on part de une besoin puis rechache puis mis en place test etc j’aimm beaucoup je vais me speciale en cyber et infrastra

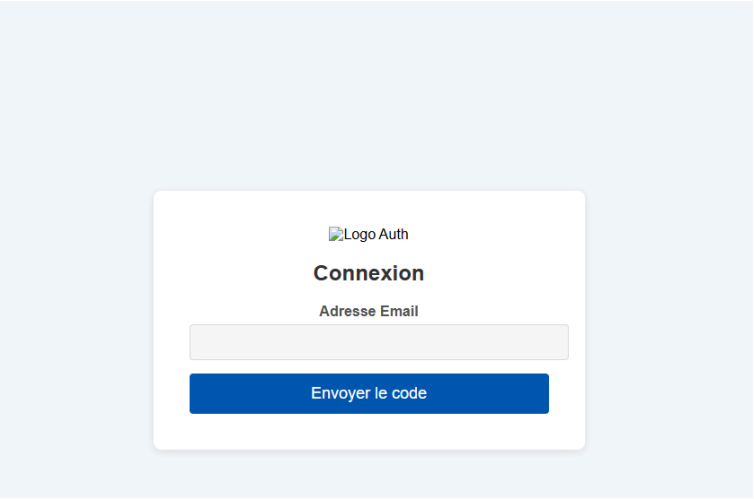
Methode 2

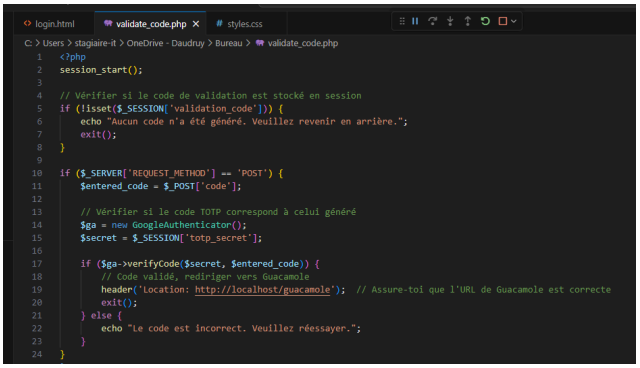
Architecture proposée :

1. **Utilisateur sur Guacamole :** L'utilisateur entre ses identifiants dans l'interface de connexion de **Guacamole**.
2. **Redirection vers un serveur Apache d'authentification externe :** Après la saisie des identifiants, **Guacamole** redirige l'utilisateur vers un serveur Apache dédié à l'authentification.
3. **Authentification par email :** Sur ce serveur, l'utilisateur saisit un code envoyé par email (authentification à deux facteurs).
4. **Redirection vers Guacamole :** Une fois le code validé, le serveur Apache redirige l'utilisateur vers **Guacamole** pour accéder à ses sessions.

En effet, je crée un serveur **Apache séparé** qui gère l'authentification à deux facteurs et redirige l'utilisateur vers **Guacamole** une fois le code validé.

```
zafar@srvauth:~$ sudo apt install apache2
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
```

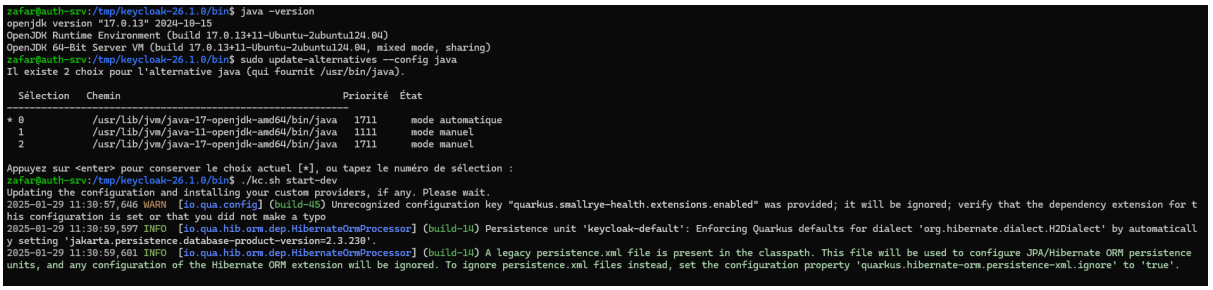




Cette solution ets tres pousé c’est
fessable avec html css et php et redrictaion des page mais cela risque de prendre le temps et je suis eau 4em semain de stage donc je tente un autre solution de ath

Methode 3

Installer Keycloak : je sui avance masi il y certain endroit ou il faut avoir ds compétence en développement



Methode 4

mettre en place l'authentification SSO via SAML entre Guacamole et Keycloak

---

1. Préparer l'environnement

- **Serveur Guacamole : Assure-toi que Guacamole est installé et fonctionne correctement.**
- **Serveur Keycloak : Assure-toi que Keycloak est installé et accessible.**

```
root@auth-srv:~# cd /tmp/
root@auth-srv:/tmp# ls
hsperfdata_root
hsperfdata_zafar
keycloak-26.1.0.tar.gz
```

```
root@auth-srv:/tmp# sudo mv /tmp/keycloak-26.1.0 /opt/keycloak
root@auth-srv:/tmp#
```

```
root@auth-srv:/opt/keycloak# cd keycloak-26.1.0/
root@auth-srv:/opt/keycloak/keycloak-26.1.0# ls
bin  conf  lib  LICENSE.txt  providers  README.md  themes  version.txt
root@auth-srv:/opt/keycloak/keycloak-26.1.0# cd bin/
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin# ls
client  federation-sssd-setup.sh  kcadm.bat  kcadm.sh  kc.bat  kcreg.bat  kcreg.sh  kc.sh
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin#
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin# ls -l ./kc.sh
-rwxr-xr-x 1 1001 118 6286 janv. 15 10:25 ./kc.sh
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin# chmod +x kc.sh
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin#
```

```
root@auth-srv:/opt/keycloak/keycloak-26.1.0/bin# ./kc.sh start-dev
Updating the configuration and installing your custom providers, if any. Please wait.
2025-01-30 09:11:07,416 WARN  [io.qua.config] (build-17) Unrecognized configuration key "quarkus.small
rye-health.extensions.enabled" was provided; it will be ignored; verify that the dependency extension
for this configuration is set or that you did not make a typo
2025-01-30 09:11:09,300 INFO  [io.qua.hib.orm.dep.HibernateOrmProcessor] (build-4) Persistence unit 'k
eycloak-default': Enforcing Quarkus defaults for dialect 'org.hibernate.dialect.H2Dialect' by automati
cally setting 'jakarta.persistence.database-product-version=2.3.230'.
2025-01-30 09:11:09,303 INFO  [io.qua.hib.orm.dep.HibernateOrmProcessor] (build-4) A legacy persistenc
e.xml file is present in the classpath. This file will be used to configure JPA/Hibernate ORM persiste
nce units, and any configuration of the Hibernate ORM extension will be ignored. To ignore persistence
.xml files instead, set the configuration property 'quarkus.hibernate-orm.persistence-xml.ignore' to '
true'.
```

✅ 1. Modifier Keycloak pour autoriser les connexions distantes

Ouvre le fichier **Keycloak.conf** :

bash — Copier — Modifier

```
sudo nano /opt/keycloak-26.1.0/conf/keycloak.conf
```

Ajoute ces lignes à la fin :

makefile — Copier — Modifier

```
http-enabled=true
http-host=0.0.0.0
hostname=10.10.10.8
http-port=8080
admin=admin
admin-password=admin
```

💡 **Explication :**

- `http-enabled=true` → Active HTTP (désactive l'obligation d'HTTPS).
- `http-host=0.0.0.0` → Accepte les connexions depuis toutes les IPs.
- `hostname=10.10.10.8` → Définit l'IP du serveur comme l'host.
- `http-port=8080` → Définit le port d'écoute.
- `admin=admin` → Force la création d'un compte admin.
- `admin-password=admin` → Définit le mot (↓)asse.

✅ 2. Contourner la restriction avec SSH (si accès distant impossible)

Si Keycloak bloque encore l'accès **depuis une autre machine**, utilise **un tunnel SSH** depuis Windows :

1️⃣ **Ouvre une invite de commande sur Windows et entre :**

powershell — Copier — Modifier

```
ssh -L 8080:localhost:8080 zafar@10.10.10.8
```

2️⃣ **Sur ton navigateur Windows**, ouvre :

arduino — Copier — Modifier

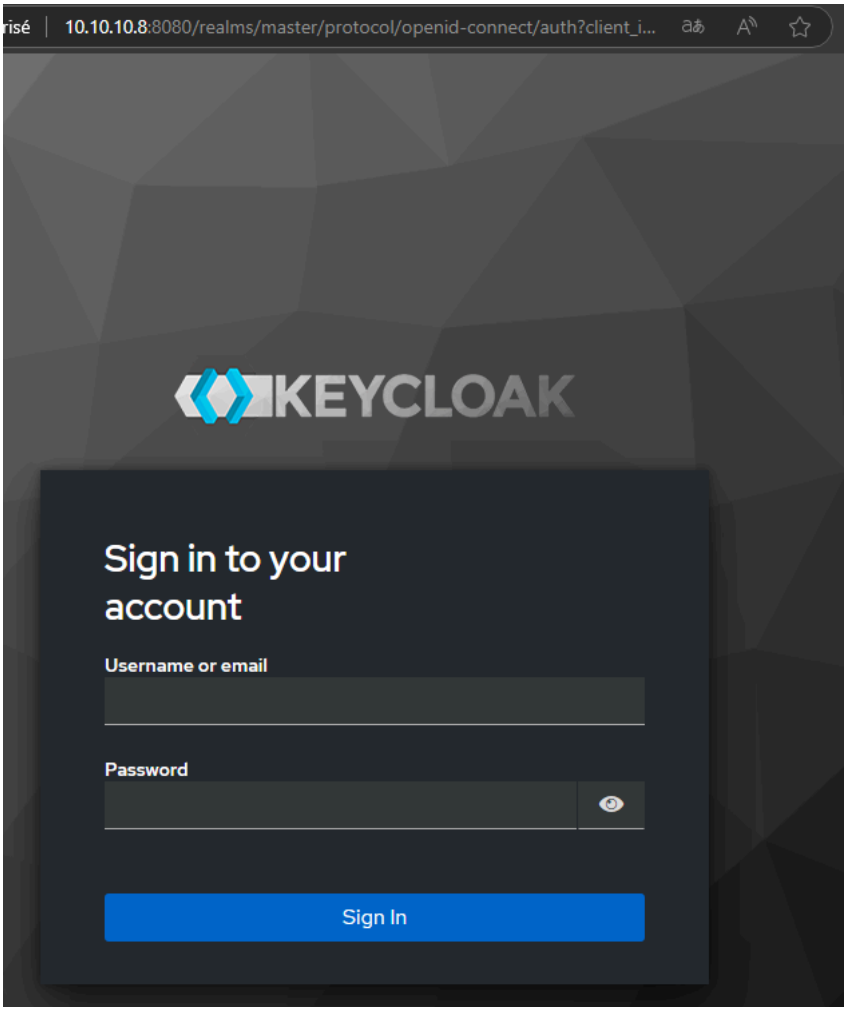
```
http://localhost:8080
```

👉 Tu seras **redirigé vers Keycloak sur le serveur distant.**

On lance notre serveur :

```
zafar@auth-srv:/opt/keycloak-26.1.0/bin$ sudo ./kc.sh start-dev
Running the server in development mode. DO NOT use this configuration in production.
2025-01-31 09:38:56,411 WARN  [io.quarkus.config] (main) Unrecognized configuration key "quarkus.small
rye-health.extensions.enabled" was provided; it will be ignored; verify that the dependency extension
for this configuration is set or that you did not make a typo
2025-01-31 09:38:56,827 INFO  [org.keycloak.url.HostnameV2ProviderFactory] (main) If hostname is speci
fied, hostname-strict is effectively ignored
2025-01-31 09:38:58,730 INFO  [org.keycloak.quarkus.runtime.storage.infinispan.CacheManagerFactory] (T
hread-5) Starting Infinispan embedded cache manager
2025-01-31 09:38:58,789 INFO  [io.agroal.pool] (JPA Startup Thread) Datasource '<default>': Initial si
ze smaller than min. Connections will be created when necessary
2025-01-31 09:38:58,884 INFO  [org.infinispan.CONTAINER] (Thread-5) Virtual threads support enabled
```

Test la Connexion à Distance



Pour le moment le firewall est désactivé :

```
zafar@auth-srv:/opt/keycloak-26.1.0/bin$ sudo ufw status
Status: inactive
zafar@auth-srv:/opt/keycloak-26.1.0/bin$
```

Résumé

1️⃣ **Modifier `keycloak.conf` pour autoriser l'accès distant.**
2️⃣ **Exécuter `kc.sh build` puis redémarrer Keycloak.**
3️⃣ **Utiliser un tunnel SSH si Windows ne peut pas accéder à Keycloak.**

Comparaison des méthodes d'authentification

Méthode	Avantages	Inconvénients	Exemples d'utilisation
OAuth2/OpenID Connect	SSO, intégration facile avec de nombreux services	Nécessite un serveur d'identité (ex. Keycloak)	Intégration avec Google, Facebook, etc.
LDAP	Intégration avec Active Directory, gestion centralisée des utilisateurs	Nécessite un serveur LDAP/Active Directory	Entreprises utilisant LDAP/Active Directory
JWT (JSON Web Token)	Authentification stateless, sécurisé, aucune session côté serveur	Gestion des jetons et des clés secrètes	Applications modernes, API sécurisées
SMS (2FA)	Sécurisé, utilisé largement	Coût des messages SMS, dépend d'une passerelle SMS	Authentification via SMS (Twilio, Nexmo)
Google Authenticator/Authy	Sécurisé, ne dépend pas d'une connexion internet	L'utilisateur doit installer une application	Utilisé avec 2FA, Authentification mobile

Methode 5

```
zafar@auth-srv:~$ apt-cache search jre
default-jre - environnement d'exécution Java standard ou compatible
default-jre-headless - environnement d'exécution standard Java ou compatible - non graphique
openjdk-17-jre - Environnement d'exécution Java OpenJDK qui utilise Hotspot JIT
openjdk-17-jre-headless - environnement d'exécution Java OpenJDK utilisant Hotspot JIT (sans affichage
)
openjdk-21-jre - Environnement d'exécution Java OpenJDK qui utilise Hotspot JIT
openjdk-21-jre-headless - environnement d'exécution Java OpenJDK utilisant Hotspot JIT (sans affichage
)
android-platform-tools-base - outils de base pour développer des applications pour le système Android
docbook-xsl - feuilles de style de conversion de fichiers DocBook XML vers différents formats
```

```
zafar@auth-srv:~$ sudo apt install -y openjdk-21-jre
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
```

```
zafar@auth-srv:/opt$ sudo wget https://github.com/keycloak/keycloak/releases/download/26.1.0/keycloak-
26.1.0.tar.gz
--2025-01-31 08:08:31--  https://github.com/keycloak/keycloak/releases/download/26.1.0/keycloak-26.1.0
.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/11125589/ff752a
ee-0a36-473d-9168-7fa9355643c6?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproductio
n%2F20250131%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250131T080831Z&X-Amz-Expires=300&X-Amz-Signa
ture=e2b32e3a69c86b391491392afe42c26620b128bb01fb1546681ad306e38dde5a&X-Amz-SignedHeaders=host&respons
e-content-disposition=attachment%3B%20filename%3Dkeycloak-26.1.0.tar.gz&response-content-type=applicat
ion%2Foctet-stream [following]
--2025-01-31 08:08:32--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/
11125589/ff752aee-0a36-473d-9168-7fa9355643c6?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releas
eassetproduction%2F20250131%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250131T080831Z&X-Amz-Expires=
300&X-Amz-Signature=e2b32e3a69c86b391491392afe42c26620b128bb01fb1546681ad306e38dde5a&X-Amz-SignedHeade
rs=host&response-content-disposition=attachment%3B%20filename%3Dkeycloak-26.1.0.tar.gz&response-conten
t-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.11
1.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 148227819 (141M) [application/octet-stream]
Saving to: ‘keycloak-26.1.0.tar.gz’

keycloak-26.1.0.tar.gz    100%[===================================>] 141,36M  74,5MB/s    in 1,9s

2025-01-31 08:08:34 (74,5 MB/s) - ‘keycloak-26.1.0.tar.gz’ saved [148227819/148227819]
```

```
zafar@auth-srv:/opt$ sudo tar xzvf keycloak-26.1.0.tar.gz
keycloak-26.1.0/version.txt
keycloak-26.1.0/conf/cache-ispn.xml
keycloak-26.1.0/README.md
```

```
zafar@auth-srv:/opt$ ls
keycloak-26.1.0  keycloak-26.1.0.tar.gz
zafar@auth-srv:/opt$
```

Line symbolique ou cas ou

```
zafar@auth-srv:/opt$ sudo ln -s keycloak-26.1.0/ keycloak
zafar@auth-srv:/opt$
```

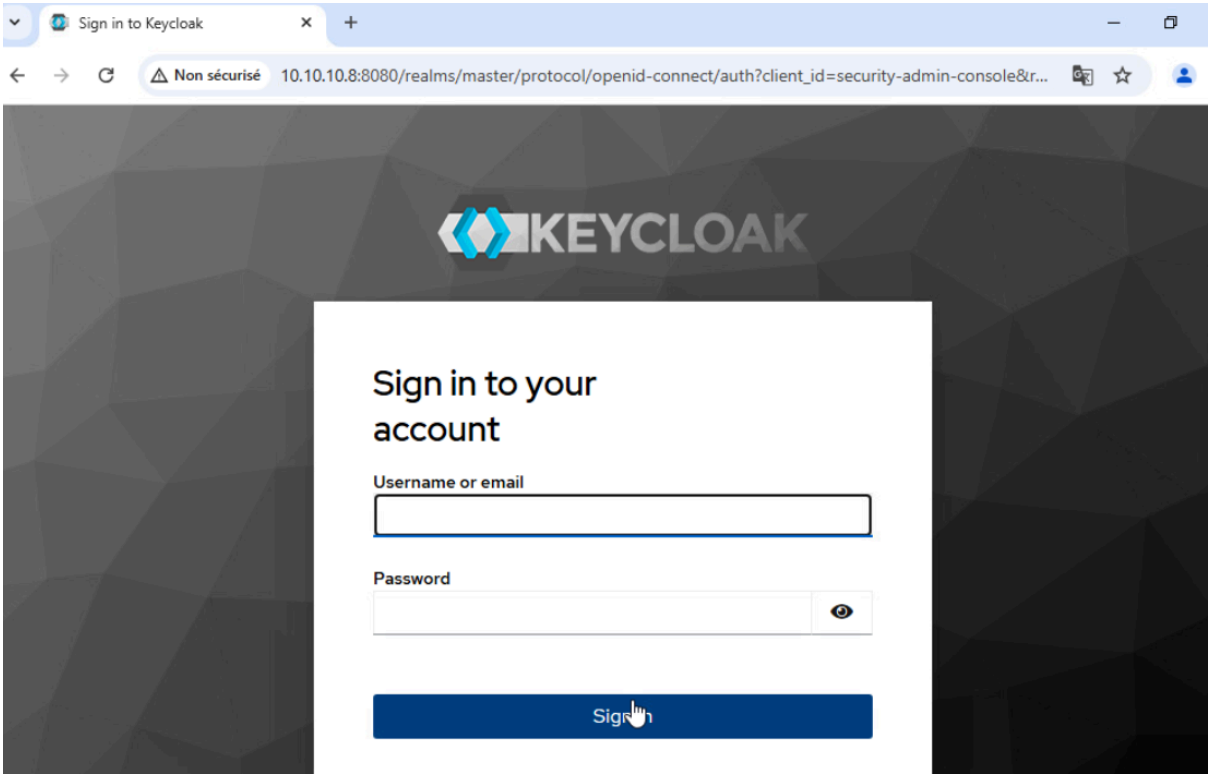
```
zafar@auth-srv:/opt$ cd keycloak
zafar@auth-srv:/opt/keycloak$ ls
bin  conf  lib  LICENSE.txt  providers  README.md  themes  version.txt
zafar@auth-srv:/opt/keycloak$ ls
bin  conf  lib  LICENSE.txt  providers  README.md  themes  version.txt
zafar@auth-srv:/opt/keycloak$ ls bin/
client  federation-sssd-setup.sh  kcadm.bat  kcadm.sh  kc.bat  kcreg.bat  kcreg.sh  kc.sh
zafar@auth-srv:/opt/keycloak$
```

Ajoute des utilisateur :

```
 GNU nano 7.2                          /opt/keycloak-26.1.0/conf/keycloak.conf
#metrics-enabled=true

# HTTP

# The file path to a server certificate or certificate chain in PEM format.
#https-certificate-file=${kc.home.dir}conf/server.crt.pem

# The file path to a private key in PEM format.
#https-certificate-key-file=${kc.home.dir}conf/server.key.pem

# The proxy address forwarding mode if the server is behind a reverse proxy.
#proxy=reencrypt

# Do not attach route to cookies and rely on the session affinity capabilities from reverse proxy
#spi-sticky-session-encoder-infinispan-should-attach-route=false

# Hostname for the Keycloak server.
#hostname=myhostname

http-enabled=true
http-host=0.0.0.0
hostname=10.10.10.8
http-port=8080
admin=admin
admin-password=admin
```

On execute build :

```
zafar@auth-srv:/opt/keycloak-26.1.0/bin$ sudo ./kc.sh build
WARNING: The following run time options were found, but will be ignored during build time: kc.hostname
, kc.http-enabled, kc.http-host, kc.http-port

Updating the configuration and installing your custom providers, if any. Please wait.
2025-01-31 09:27:45,987 WARN  [io.qua.config] (build-5) Unrecognized configuration key "quarkus.smallr
ye-health.extensions.enabled" was provided; it will be ignored; verify that the dependency extension f
```


Sign in to Keycloak
Non sécurisé
10.10.10.8:8080/realms/master/protocol/openid-connect/auth?client_id=security-admin-console&r...
KEYCLOAK
Sign in to your account
Username or email
Password
Sign

Étapes pour mettre en place l'authentification SSO via SAML entre Guacamole et Keycloak

---

1. Préparer l'environnement

- **Serveur Guacamole : Assure-toi que Guacamole est installé et fonctionne correctement.**
- **Serveur Keycloak : Assure-toi que Keycloak est installé et accessible.**

---

2. Installer l'extension SSO pour Guacamole

a. Télécharger l'extension SSO/SAML pour Guacamole :

- **Télécharge le fichier `guacamole-auth-sso-saml-1.4.0.jar` à partir du dépôt Apache ou de la source officielle.**

b. Installer l'extension dans Guacamole :

- **Copie le fichier JAR dans le répertoire extensions de Guacamole, généralement situé dans `/etc/guacamole/extensions/`.**

bash

Copier

```
sudo cp guacamole-auth-sso-saml-1.4.0.jar /etc/guacamole/extensions/
```

- **Assure-toi que le serveur Guacamole est arrêté avant d'ajouter l'extension.**

c. Redémarrer Guacamole :

bash

Copier

```
sudo systemctl restart guacamole
```

---

3. Configurer Keycloak comme fournisseur d'identité (IdP)

a. Créer un Realm dans Keycloak :

- **Connecte-toi à l'interface d'administration de Keycloak.**
- **Crée un Realm spécifique pour Guacamole (par exemple, `Guacamole-Realm`).**

b. Créer un client dans Keycloak pour Guacamole :

- **Dans Keycloak, va dans Clients et crée un client pour Guacamole.**
- **Le type de client doit être OpenID Connect ou SAML.**

Exemple de configuration pour Keycloak :

- **Client ID : `guacamole-client`**
- **Client Secret : Génére un secret pour ce client.**
- **Root URL : L'URL de ton serveur Guacamole (par exemple `http://guacamole-server/guacamole`).**

c. Configurer les paramètres SAML dans Keycloak :

- **Active le protocole SAML pour ce client.**
- **Assure-toi que les métadonnées SAML (URL d'IdP, entité IdP, etc.) sont bien définies.**
- **Obtiens l'URL du SSO (Single Sign-On) pour intégrer dans Guacamole.**

---

4. Configurer Guacamole pour utiliser l'authentification SSO via SAML

a. Modifier le fichier `guacamole.properties` :

Ouvre le fichier `guacamole.properties` et ajoute les configurations pour utiliser SAML comme mécanisme d'authentification.

Exemple de configuration pour Guacamole dans `guacamole.properties` :

ini

Copier

```
# Activer l'authentification SSO via SAML
```

```
auth-provider:
net.sourceforge.guacamole.auth.sso.SSOAuthenticationProvider


# URL du fournisseur d'identité SAML (Keycloak)

sso-saml-idp-url:
https://<keycloak-server>/realms/Guacamole-Realm/protocol/saml


# ID de l'entité du fournisseur SAML

sso-saml-idp-entity-id: Guacamole-Realm


# Configuration du client SAML dans Keycloak

sso-saml-client-id: guacamole-client


# Secret du client SAML dans Keycloak

sso-saml-client-secret: <client-secret>
```

b. Redémarrer Guacamole pour appliquer les changements :

bash

Copier

```
sudo systemctl restart guacamole
```

---

5. Tester l'authentification SSO

1. **Accéder à Guacamole : Ouvre un navigateur et accède à Guacamole via l'URL configurée (par exemple `http://guacamole-server/guacamole`).**
2. **Redirection vers Keycloak : L'utilisateur sera automatiquement redirigé vers Keycloak pour s'authentifier (s'il n'est pas déjà authentifié).**
    - **L'utilisateur saisit ses identifiants dans Keycloak.**
3. **Authentification réussie : Une fois l'utilisateur authentifié, Keycloak génère une assertion SAML et redirige l'utilisateur vers Guacamole avec une session active.**
4. **Accès à Guacamole : Une fois l'utilisateur authentifié, il peut accéder aux différentes connexions distantes (RDP, SSH, etc.) configurées dans Guacamole sans avoir à saisir de nouveau ses identifiants.**

---

6. Vérification des logs et du bon fonctionnement

a. Vérifier les logs de Guacamole :

Si l'authentification échoue, vérifie les logs de Guacamole pour détecter toute erreur :

bash

Copier

```
sudo tail -f /var/log/guacamole/guacamole.log
```

b. Vérifier les logs de Keycloak :

Si l'authentification échoue côté Keycloak, vérifie les logs de Keycloak pour toute erreur dans l'authentification SSO.

Résumé des étapes

1. **Installer Guacamole et l'extension SSO via SAML.**
2. **Configurer Keycloak comme fournisseur d'identité SAML.**
3. **Configurer Guacamole pour utiliser Keycloak via SAML pour l'authentification SSO.**
4. **Tester l'intégration et vérifier que l'utilisateur est redirigé vers Keycloak pour se connecter et est ensuite redirigé vers Guacamole avec une session active.**