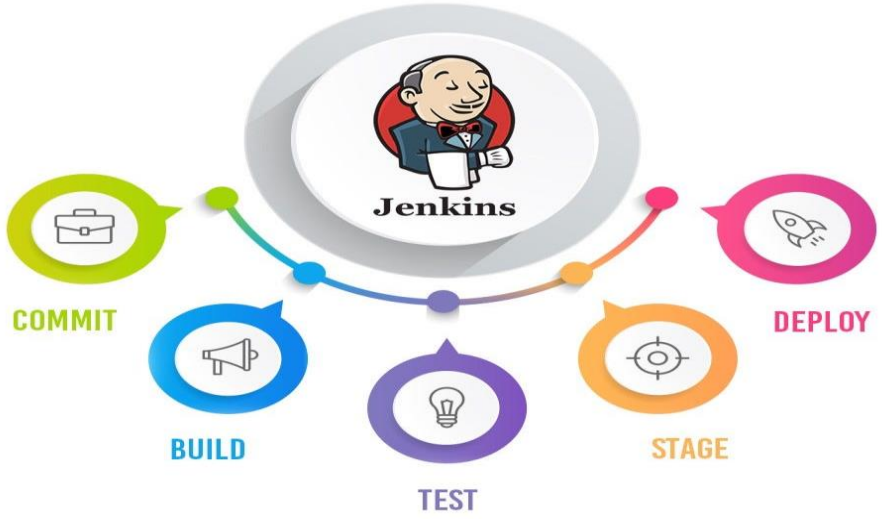
Jenkins



Le projet :

1- Créer une application Web en **Html/Css et JavaScript** et une api en ASP.NET Web API 6.

2- Installer et configurer le serveur web **Apache2** sur le serveur linux **Debian 192.168.153.131** afin de pouvoir héberger l'app web.

3- Installer et configurer le serveur web **Nginx** pour héberger l'app web sur le serveur **CentOS 192.168.153.133.**

4- Installer et configurer le serveur web **IIS** sur le contrôleur de domaine **Windows 192.168.153.130.**

5- Installer et configurer un serveur **Jenkins** sur **192.168.153.132** qui permettra de déployer et intégrer de façon continue les sources de l'application web sur les différents nœuds (Linux **Debian, CentOS** et **Windows)**.

6- Faire la même chose que (5) mais déployer sur des workers d'un node master **Kubernetes**.

7- Configurer **Jenkins** pour déployer automatiquement sur le serveur linux **Debian et CentOS** les sources de l'application au moment du merge sur la branche master de **GitHub**.

8- Ne pas installer docker sur le serveur linux cible **Debian** mais d'ajouter un plugins **Docker** dans **Jenkins** afin de Builder le **Dockerfile** ou même de lancer le conteneur.

Rédigé par : Artur Lambo DevOps engineer

9- Créer un serveur de base de données **MSSQL** définit dans le contrôleur de domaine **192.168.153.130**.
10- Créer une image pour une **API** web en NET6.0 définit dans un service **Docker**.

11- Installer et configurer l'utilitaire **Traefik** permettant de concilier dans le même réseau les deux services **Docker** (**Api** et **App**).

- Caching
- Répartition de charges entre services docker (notamment pour l'api .net6)
- SSL/TLS (on va le faire via haproxy)

12- On va utiliser **Traefik** de deux façons :
- Avec un fournisseur docker
- Avec un fournisseur kube

13- Installer et configurer un **DOZZLE** permettant de lire les logs des applications isolées dans **Docker**.

14- Utiliser **Ansible** notamment les rôles ansible pour déployer les configurations automatiquement sur le serveur Linux **Debian et CentOS et Windows** (la création de site web, pool d'application, chemin virtuel et physique, installation de chocolatey, node js pour les sites IIS**)**.

15- Installation de **Grafana** pour contrôler et monitorer les **VM.**

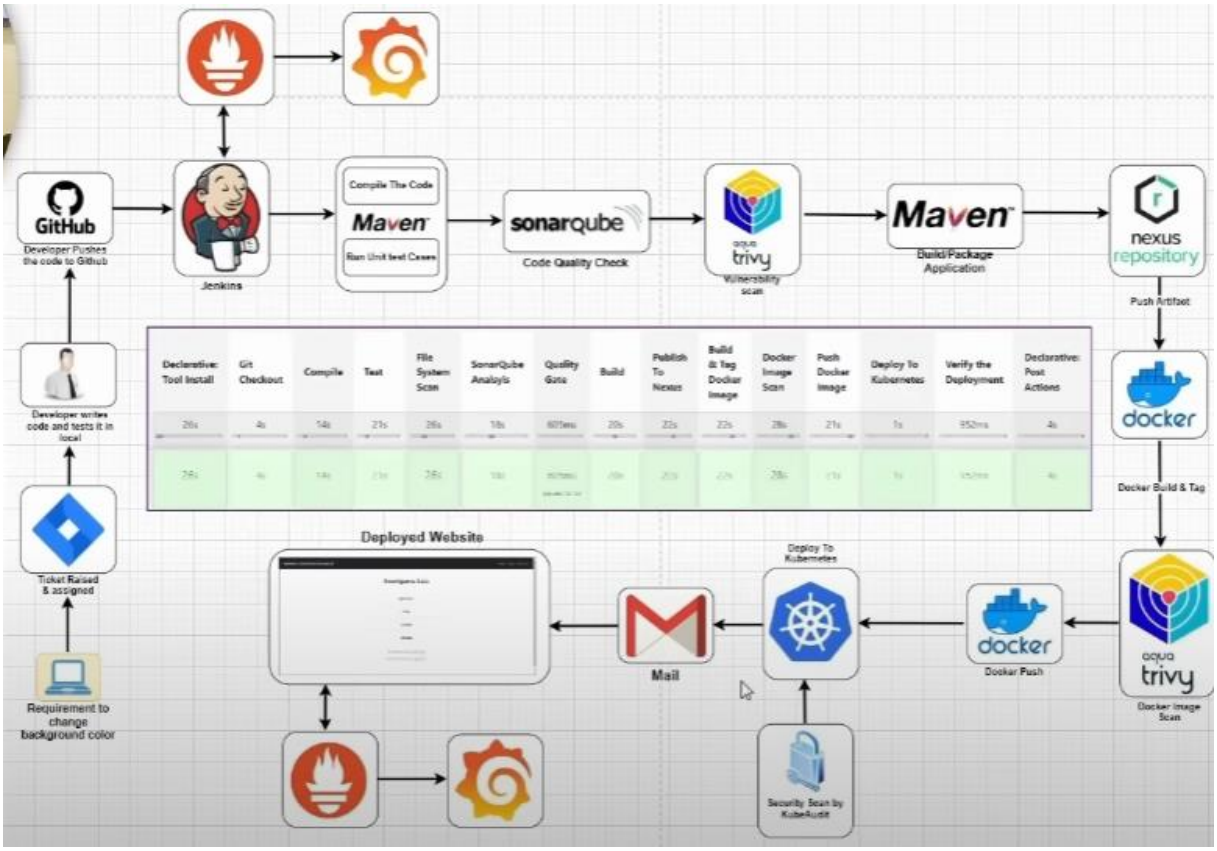
16- Plugger **prométhéums** , **Traefik** et **Grafana** ???

Remarque générale pour la configuration d'un service

```
Quand on configure un service sous linux on doit se rassurer

-   L'exécutable de ce service
-   S'il existe un compte de service associé à ce service
-   Des fichiers de logs et leur emplacement
-   De la configuration du fichier de lancement de ce service
-   Des fichiers de configuration du service en question
-   Du port d'écoute de ce service
-   Des dépendances liées à ce service
-   Des permissions sur les différents fichiers liés à ce service et
le propriétaire (généralement le service lui-même )
```


Rédigé par : Artur Lambo DevOps engineer

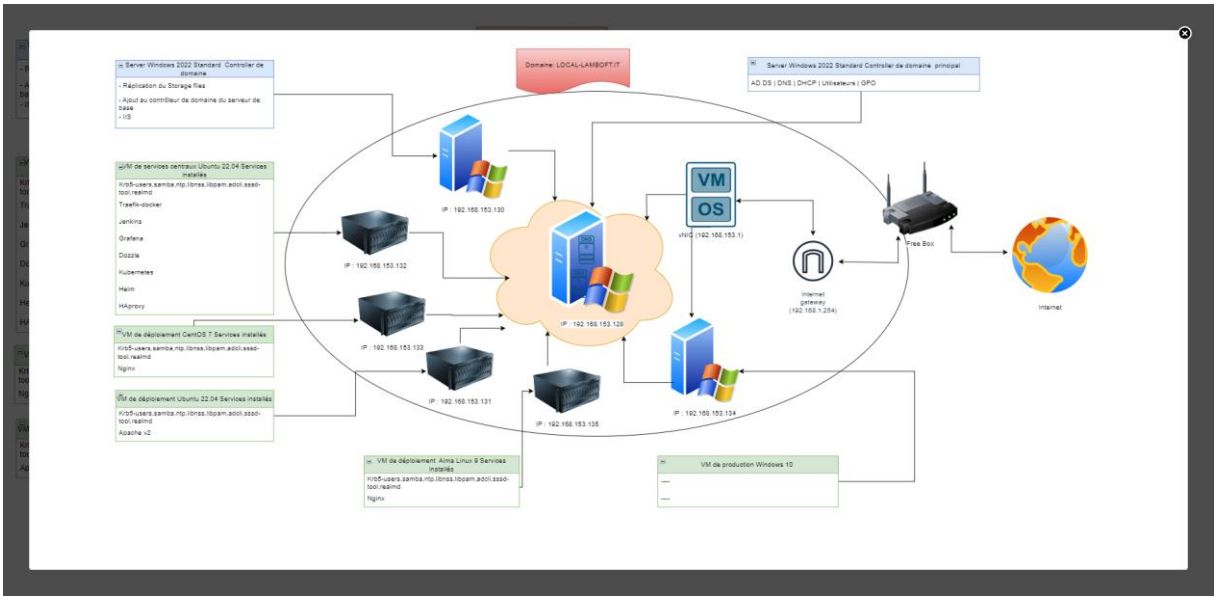


Préparation des serveurs de développement et de production pour le projet

On va se basé sur le schéma suivant pour constituer notre propre écosystème de développement et de déploiement.

Nous allons alterner entre administration de OS et réseau au sein de ces OS. Plusieurs concepts vont être évoqués notamment : DNS , DHCP, ADDS, Passerelle par défaut et Proxy.

I Architecture system



II Sécuriser notre serveur Jenkins

- Pouvoir accéder à notre serveur jenkins via https et configurer un HAproxy pour de la redirection de port sur le port de jenkins aussi pouvoir centraliser les certificat SSL
- Gérer la création des utilisateurs (on veut que les utilisateurs configurer dans l'AD puissent se connecter à notre jenkins)

i. Haproxy et Jenkins

Utilisation combinée de HAProxy et Jenkins

1. **Distribution de la charge des builds :**
   - Si vous avez plusieurs nœuds Jenkins pour exécuter des builds, HAProxy peut équilibrer la charge entre ces nœuds pour assurer une distribution uniforme des tâches de construction.
2. **Haute disponibilité de Jenkins :**
   - Configurez HAProxy pour surveiller plusieurs instances Jenkins. Si une instance tombe en panne, HAProxy peut rediriger le trafic vers une autre instance pour assurer une disponibilité continue.
3. **Sécurisation des communications :**
   - Utilisez HAProxy pour gérer le SSL termination, en déchargeant ce travail des serveurs Jenkins. Cela permet également de centraliser la gestion des certificats SSL.
4. **Reverse Proxy :**
   - HAProxy peut agir comme un reverse proxy pour Jenkins, permettant de cacher les détails de l'infrastructure interne et de renforcer la sécurité.

Brainstorming

On va créer un seconde nœud linux sur la 192.168.153.133 afin d'expérimenter la distribution de charge des builds via haproxy.

C'est quoi une instance sur jenkins et comment gérer haproxy pour la surveillance des instances sur jenkins ?

III Création d'agents

Pourquoi ?

- La scalabilité
- La spécialisation
- L'isolation

Sur le Master à savoir la **vm** sur laquelle on a installé Jenkins

- Vérifier si un compte de service jenkins n'est pas déjà installé

getent passwd jenkins

- Crée une clé **ssh** pour ce compte de service
- Changer si nécessaire le format de la clé SSH
- Rendre le compte de service jenkins propriétaire du répertoire /var/lib/jenkins

- Rajouter ceci dans visudo

```
# Compte de service jenkins créé auto au moment de l'installation
jenkins ALL=(ALL) NOPASSWD:ALL
```

i. Création d'agent linux sur la 192.168.153.131 FQDN (vms-001-ubuntu.local-lamboft.it)

a) Sur l'agent slave en question on doit :

- ❖ Installer un **jdk** avec la même version que celle du master node de préférence.

- ❖ Créer un utilisateur Jenkins (**jenkins_linux_slave**) et un fichier .ssh/authorized_keys

```
jenkins_linux_slave@vms-001-ubuntu:~$ ls -ld .ssh/authorized_keys
-rw------- 1 jenkins_linux_slave jenkins_linux_slave 744 Jun 12 11:18 .ssh/authorized_keys
jenkins_linux_slave@vms-001-ubuntu:~$
```

- ❖ Copier la clé publique **ssh** de l'utilisateur Jenkins du node master dans ce fichier

b) Sur l'ihm de **jenkins**

Il faut :

- ❖ Configurer les crédentials pour le slave node

Si l'ID n'est pas précisé alors jenkins lui affectera un ID aléatoire .

Username : C'est le nom d'utilisateur qui a été sur le nœud slave devant gérer le repo Workspace

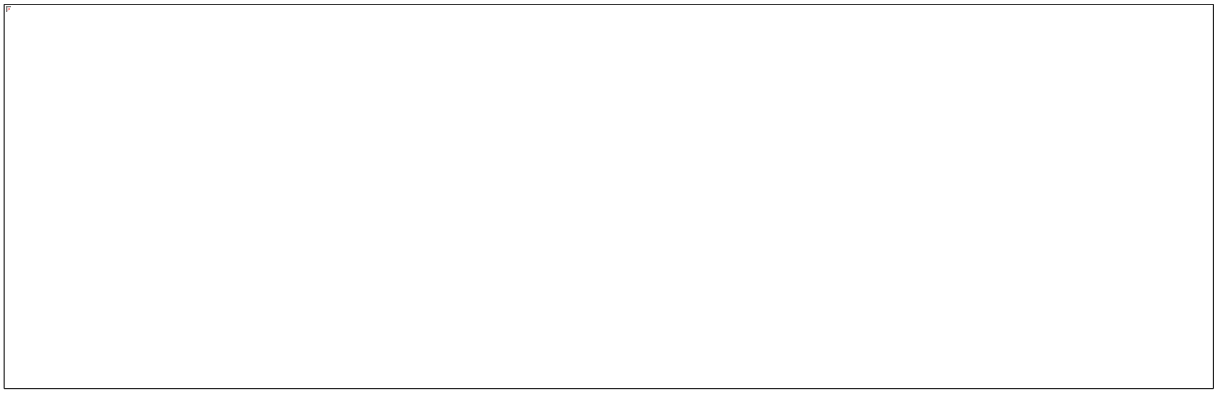
On renseigne la clé privée du compte de service jenkins du nœud Master.

- ❖ Création du nœud slave

Le nombre d'exécution doit être tributaire du nombre de processeur de la machine

L'étiquette n'est rien d'autre qu'une information nous permettant de savoir quel est l'OS sur lequel on souhaite déployer en occurrence ici on a Linux.

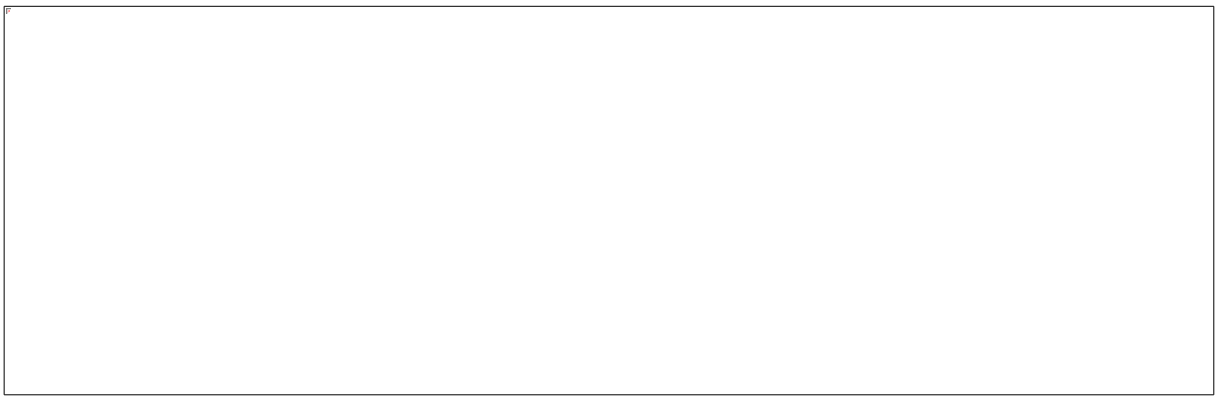
Workspace : C'est le repo dans lequel nos confs d'applications seront logés.



Type de lancement c'est du SSH

Host : On peut soit préciser le nom de la **vm** soit l'IP de celle-ci (de préférence l'IP de la VM dans le cas où il existerait tout problème avec la résolution de nom de domaine).

On ajoute les fameux crédentials créés à l'étape précédente.

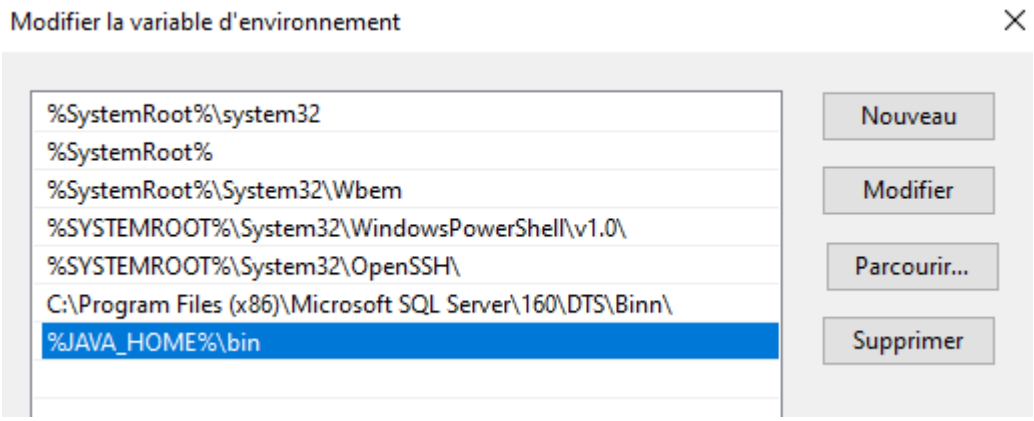


Il est nécessaire d'ajouter le chemin de l'exécutable java installée sur la machine slave

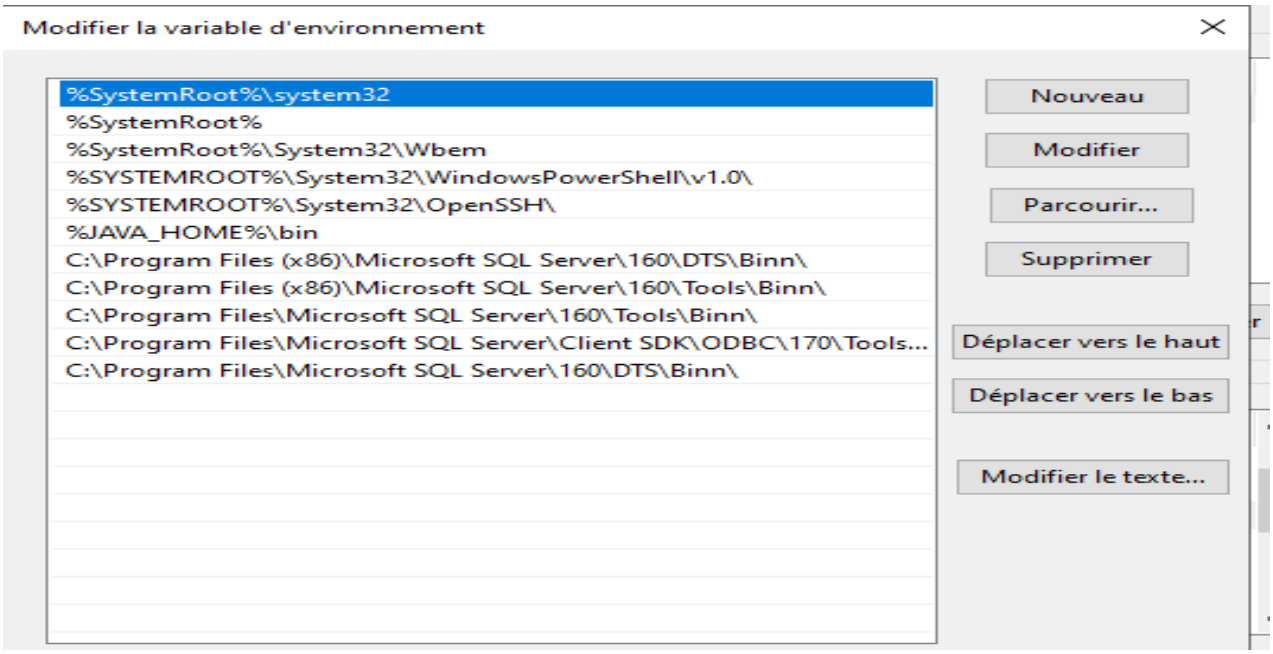
ii. Création d'agent Windows sur la 192.168.153.130 FQDN (srv-prod.local-lamboft.it)

Même procédé que pour le **Linux**

- Utilisateur ayant le droit admin sur la machine Windows
- Créer un fichier **authorized_keys** dans son **.ssh**
- Rajouter la clé publique du user jenkins du nœud maitre
- Déclarer le port 222 dans le fichier **ssh_config**
- Activer **OpenSSH** sur la machine vérifier les règles de pare-feu afin de laisser passer le trafique sur le port 222
- Installer le **jdk** avec la même version que celle du nœud maitre
- Rajouter son binaire dans les variables d'environnements système de la machine



L'image ci-dessus est la variable **JAVA_HOME** dans le **PATH**



- Se rassurer que sur le nœud maitre, l'adresse IP et la clé ssh publique des machines Windows/linux(Slaves) sont bien présentes dans les **know_host** du nœud maitre(192.168.153.132).

```
jenkins@vms-002-Server:~$ ssh-keyscan -p 222 192.168.153.130 .ssh/known_hosts
getaddrinfo .ssh/known_hosts: Name or service not known
getaddrinfo .ssh/known_hosts: Name or service not known
getaddrinfo .ssh/known_hosts: Name or service not known
getaddrinfo .ssh/known_hosts: Name or service not known
getaddrinfo .ssh/known_hosts: Name or service not known
# 192.168.153.130:222 SSH-2.0-OpenSSH_for_Windows_8.1
[192.168.153.130]:222 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC3hJrZ+AaprzpQ7+xGfFbblwRzfmQq1Zhu9vddGG71qsN4nUQJMmqjnmoFqK4Ca
porifUdYL84tY3Hp5u4X916SEFIf46xzJSB8lZMvuzrQ5NFVY9CPYLz/aLQ2cNLrmFieV9rtvst4P1fjZrj/l0IAWay+Y+nngnwtUXN0mIBuQG0p5VMYO/MpmUIw
HHdEu/DI4hDauWNd+V7bFGmC33hyrRHm0YEpGiEq34lMDR0Vtz61a6jDx7zD8OPKq15M+vcVOMWlLfdnddsW1bjjU0zy8IfWlOX1F2yskTdxFOxAzvodz/4CfTKL
# 192.168.153.130:222 SSH-2.0-OpenSSH_for_Windows_8.1
[192.168.153.130]:222 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBMr95NjbpuQrncAyE7Vy0FgJNchzt3
# 192.168.153.130:222 SSH-2.0-OpenSSH_for_Windows_8.1
[192.168.153.130]:222 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICA78sd5Ak2nLj5CIMh082S7wjQY48qcYaDu4uPrMJKt
# 192.168.153.130:222 SSH-2.0-OpenSSH_for_Windows_8.1
# 192.168.153.130:222 SSH-2.0-OpenSSH_for_Windows_8.1
```

Configuration du **slave** pour la gestion de l'application Web.

On veut pouvoir déployer une application en **html css** et **js** sur ce node dans un emplacement spécifique afin de pouvoir l'exécuter sur **IIS**.

Deux méthodes s'offrent à nous

- Configuration manuelle
- Configuration via Ansible

On va lister l'ensemble des étapes pour la configuration manuelle ensuite on va automatiser via Ansible

- ❖ L'installation des outils pour la compilation et le déploiement de l'application
- ❖ La configuration de IIS pour la création (pool d'application, site web et chemin physique pour les sources de l'application)

Configuration manuelle

- Téléchargement de chocolatey

```
Set-ExecutionPolicy Bypass -Scope Process -Force; `
[System.Net.ServicePointManager] :SecurityProtocol = [System.Net.ServicePoint-
Manager]::SecurityProtocol -bor 3072; `
iex ((New-Object System.Net.WebClient).DownloadString('https://community.cho-
colatey.org/install.ps1'))
```

- Installation de node js via chocolatey (préciser une version stable de node js)

```
choco install nodejs-lts --version=18.20.4
```

Plus besoin d'installer npm la version LTS de node l'installe aussi.

Plus besoin de configurer les variables d'environnements chocolatey le fait déjà.

On vérifie la version

```
node -v
npm -v
```

-

iii. Rattaché un projet à un nœud esclave

Dans le pipeline qui gère le déploiement du code source sur un node esclave jenkins, juste rajouter le nom de l'agent slave et si nécessaire un label identifiant cet agent

Exemple :

```
pipeline {
  agent { label 'Linux' }
  environment {
    WORKSPACE_DIR = "${env.WORKSPACE}/Dotnet-Api-TasksManagement"
    API_DIR = "${WORKSPACE_DIR}/TasksManagement_API"
    APP_NAME = 'TasksManagement_API'
    scannerHome = tool 'SonarQube-Scan-SourceCode'
  }
}
```

scannerHome : C'est le nom de notre scanner SonarQube définit dans les paramètres systèmes de Jenkins.

```
    scannerHome = tool 'SonarQube-Scan-SourceCode'
```

WORKSPACE_DIR : C'est le répertoire parent de Jenkins sur le slave + le nom du pipeline

NB : Autant de pipeline autant de nom de répertoire créé dans "${env.WORKSPACE}/

```
    WORKSPACE_DIR = "${env.WORKSPACE}/Dotnet-Api-TasksManagement"
```

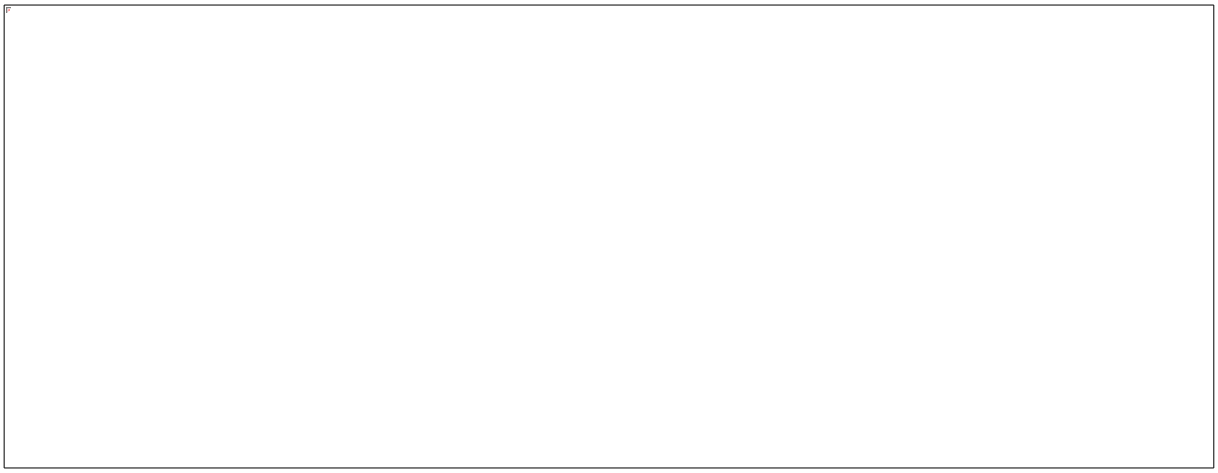
iv. Joindre les utilisateurs Active directory à Jenkins

Pour ce faire dans la liste des plugins Jenkins il va falloir rajouter :

- Le plugins active directory

- Le plugins Rule based on authorization

Après avoir rajouté ces plugins ci-haut et redémarrer jenkins pour une nouvelle prise en charge. On doit pouvoir configurer l'accès à l'AD sur le serveur.



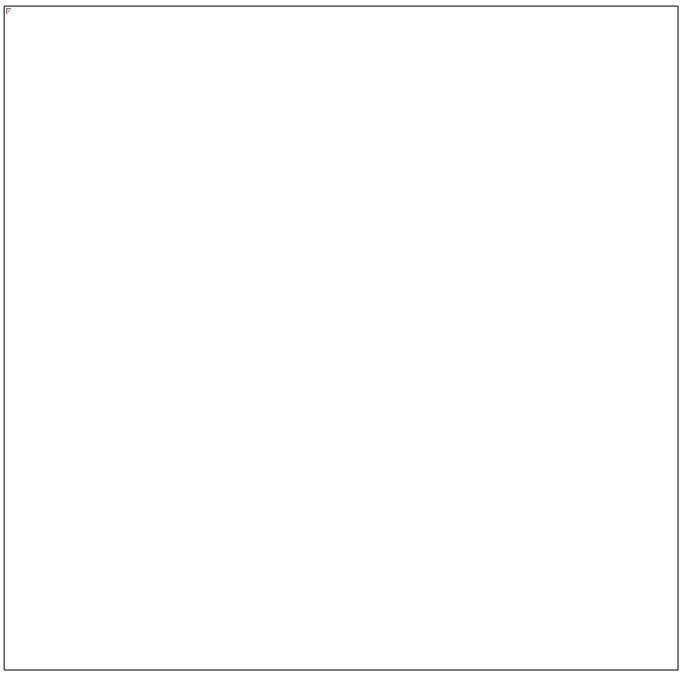
Domain Controller : il s'agit là du domaine active directory + le port LDAP (389) LDAPS (638)

bind password : C'est le mot de passe administrateur du serveur AD

Autorisations : on choisit l'autorisation basée sur les rôles (plugins qu'on a préalablement installés).

Dans la gestion des rôles, on a rajouté un nouveau rôle `developpers` (non-admin)

Avec leur ACL



Dans l'assignation des rôles on a un ensemble de groupes et d'utilisateurs ne pouvant qu'avoir deux rôles : developpers et admin

Anonymous : Si on octroie un rôle à cet utilisateur il pourra d'accéder directement à Jenkins sans même devoir s'authentifier.

NB : ne jamais lui attribuer un rôle.

Authenticated Users : On lui que dès lors qu'un utilisateur est authentifié on devrait lui attribuer le rôle de developpers.

Nb : Evidemment si l'utilisateur est admin il aura les droits admin.

jenkins-admin : Il s'agit ici d'un groupe d'utilisateurs que l'on a créé dans l'AD. En possédant le rôle admin tous les utilisateurs AD rattachés à ce groupe dans l'AD auront les droits admin sur Jenkins.

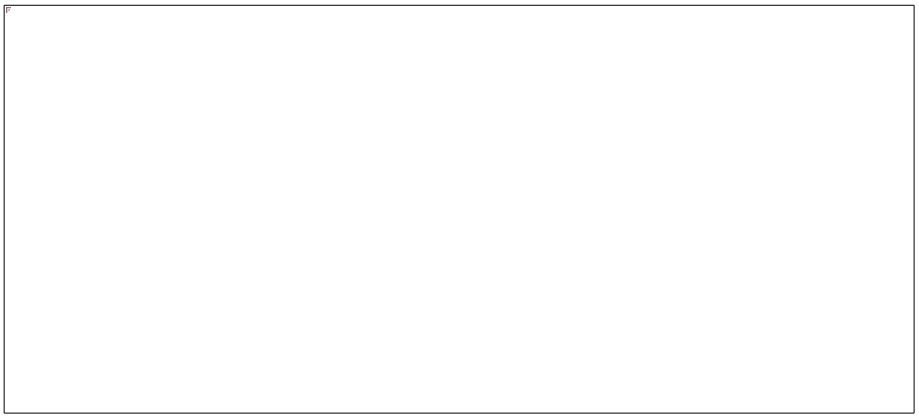
jenkins-developers : Il s'agit ici d'un groupe d'utilisateurs que l'on a créé dans l'AD. En possédant le rôle developpers tous les utilisateurs AD rattachés à ce groupe dans l'AD auront les droits developpers sur Jenkins.

On va :

- Créer un compte de service dans l'active directory pour la gestion des services Jenkins (nom : jenkinscpteservices, pass : password$2)
  On ne peut pas changer son mot de passe et il n'expire jamais

On va donner les privilèges admin à tous les utilisateurs du groupes administrateurs créé dans l'AD.

La commande id jenkinscpteservice



Il nous dit que cet utilisateur est un utilisateur du domaine AD (la machine a joint le domaine AD)

L'utilisateur appartient à des groupes du domaine notamment les groupes (BUILTIN\users et groupes administrateurs).

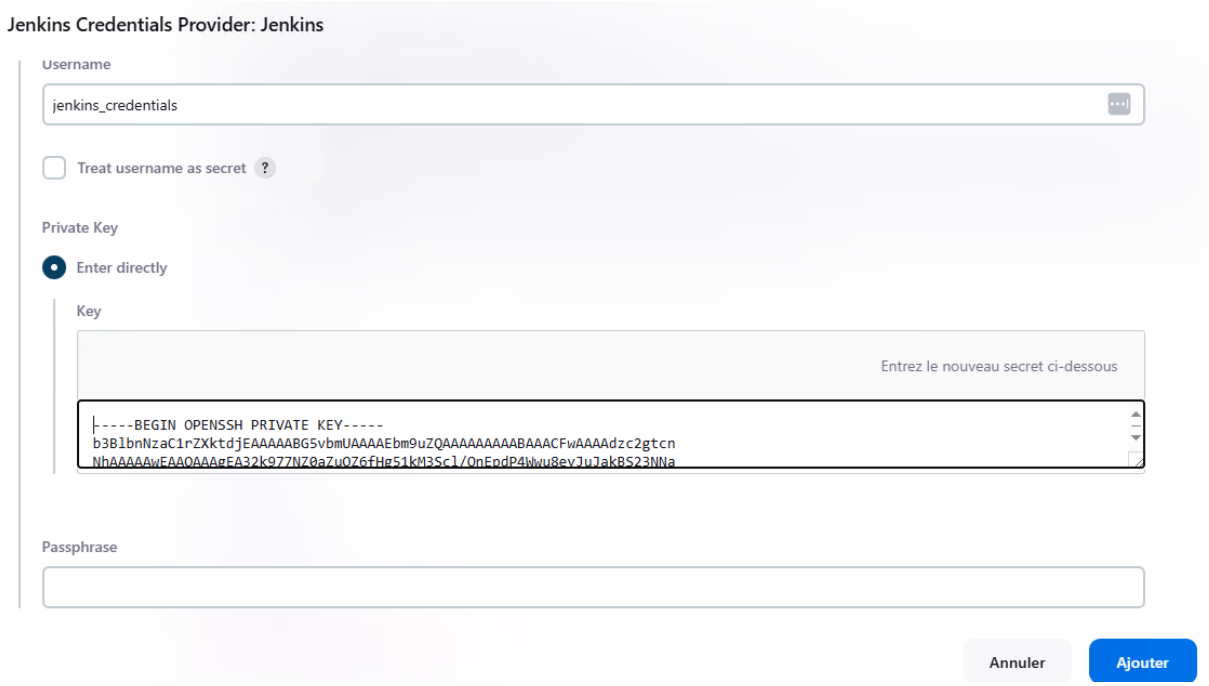
- Le node d'exécution est déjà rattaché au domaine active directory
- On va créer un repo personnel pour l'utilisateur et se rassurer que le repo appartienne bien à l'user

- On va changer les permissions et les droits du repo de Jenkins sur le serveur /var/lib/Jenkins (rendre le compte de service propriétaire de ce repo chown et chmod -R 755)

Mode de fonctionnement des nœuds avec Jenkins

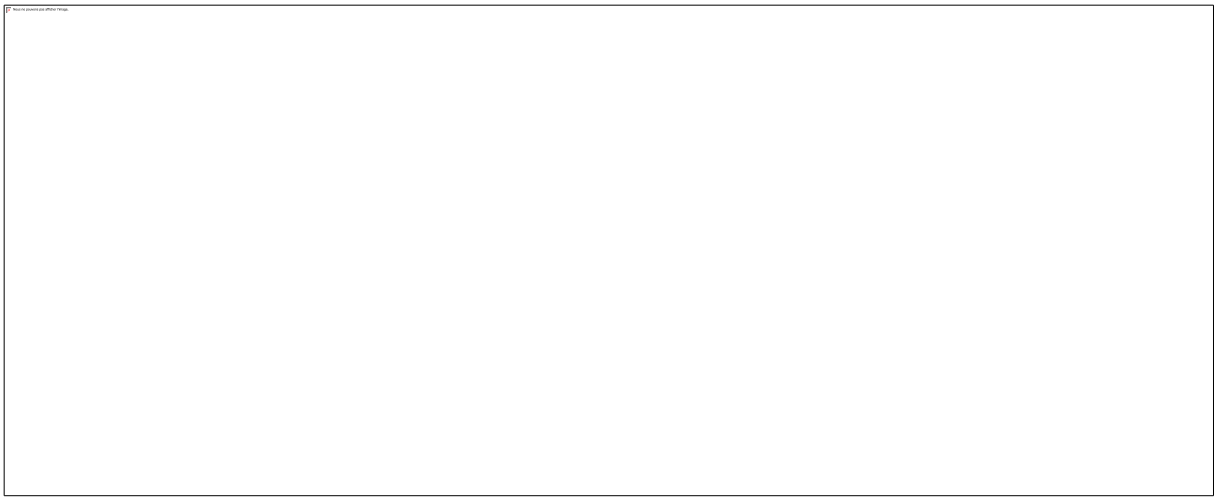
- On va créer un nœud Maitre sur le serveur vms-002-Server
- Définir un agent SSH qui assure la communication entre le master et les agents
- Créer un agent Windows(192.168.153.130) et un agent Linux (192.168.153.131)
-

Sur le node Master d



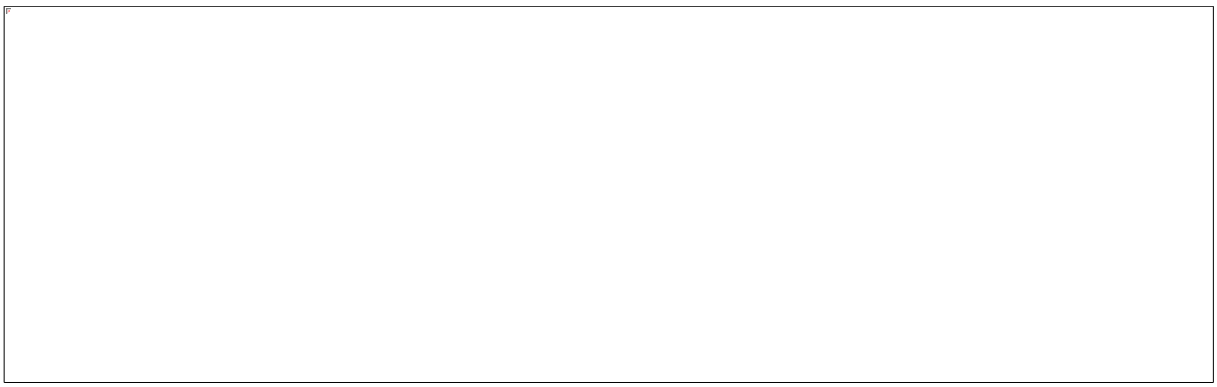
Sur le slave linux :

- Installer java jdk sur le slave (De préférence la version que sur le Master)
- Il faut préciser le **path** de l'exécutable java

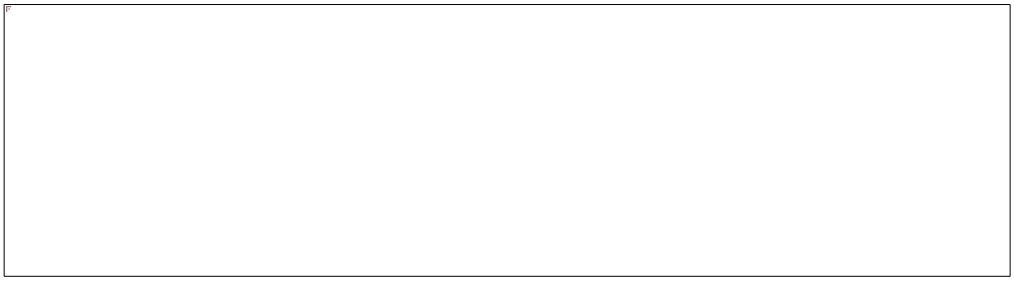


Sur le slave Windows :

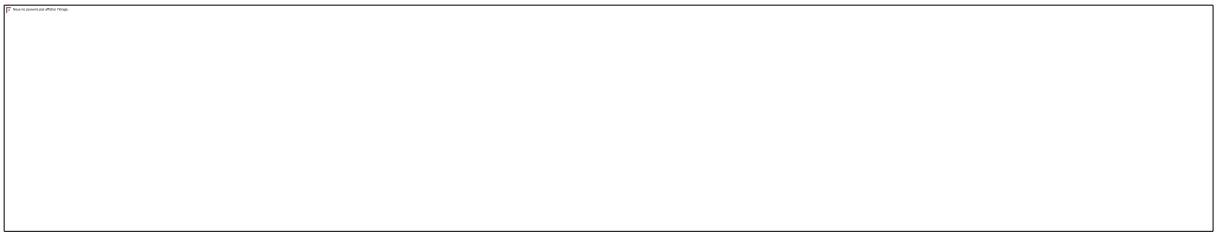
- Installation de java jdk
- Rajout de l'exécutable java dans les variables d'environnement systèmes de la machine en question
- Sur jenkins préciser le chemin de l'exécutable



contrôleur	Linux (amd64)	Synchronisé	0 B	72,50 GiB	72,50 GiB	0ms	
Slave-linux-vms-001-lts	Linux (amd64)	Synchronisé	0 B	10,21 GiB	10,21 GiB	1297ms	
Slave-windows-srv-prod		N/A	N/A	N/A	N/A	N/A	



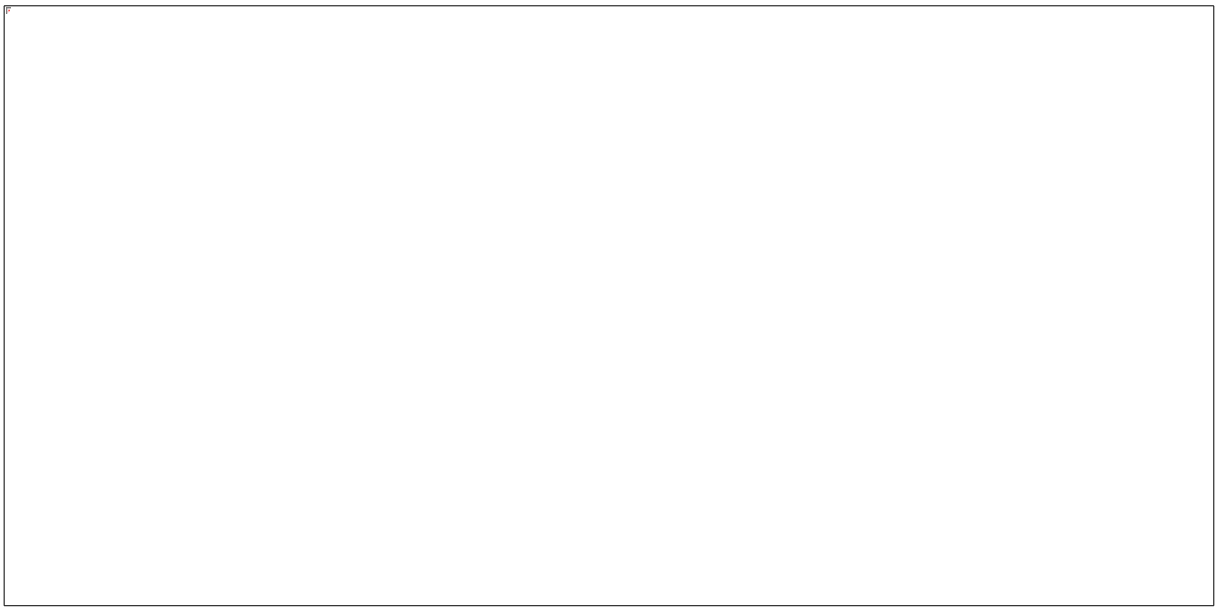
Problèmes rencontrés



0 – veut dire qu'on a installé des agents sur lesquelles les jobs seront exécutés.

>0 – veut dire qu'on souhaite exécuter les jobs sur le node master).

v. Installation de SonarQube



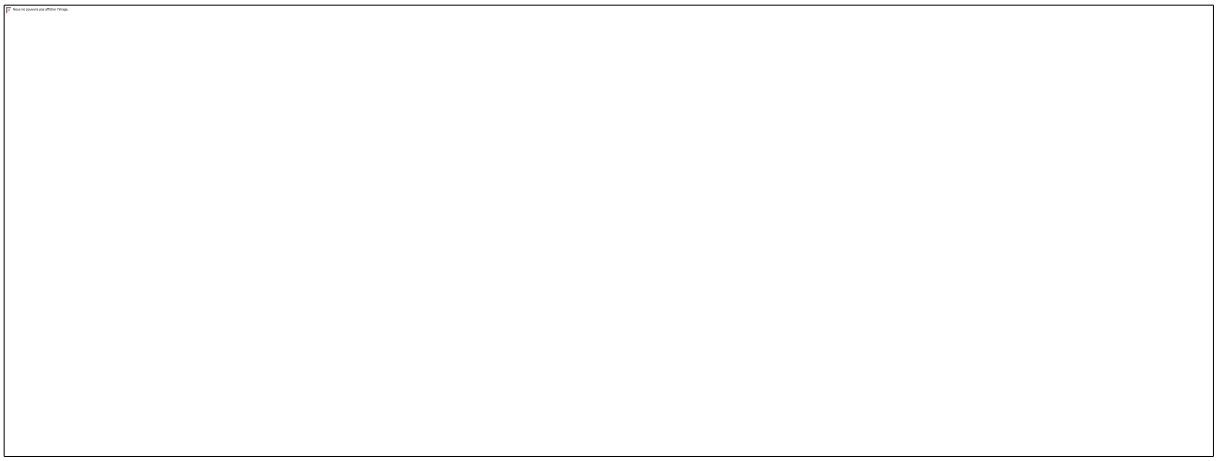
a) Sur le node master (configuration sur l'interface jenkins)

- Installation du plugins sonarQube scanner sur jenkins

- Ajout du plugin Git-plugins si ce n'est pas déjà fait

Dans Tools

- Aller sur **installations sonarqube scanner**



Name : Représente le nom du sonarQube server qu'on va renseigner dans **système** pour identifier le sonarqube scanner a utilisé.

Jenkins > système

b) Sur le node master (configuration sur la VM)

Tout d'abord on doit rappeler que SonarQube est constitué de 03 Sever

- Scanner (sur Jenkins)
- SonarQube Server (un serveur web qui héberge sonar – > nginx, apache)
- Sonar DataBase (Oracle 21 C ou 19 C , PostgreSQL 11 à 16 ou MS SQL 2022 ou 2016)

Entre le scanner et le serveur Sonar on a la possibilité d'analyser et de ressortir un rapport.

SonarQube Server : Est constitué de 03 composants

- Server Web
- Computer Engine
- Server Search (Elastic Search)

Dans **/etc/elasticsearch/elasticsearch.yml**

```
# ---------------------------------- Cluster -----------------------------------
cluster.name: Tasks-Management-ELK

# ------------------------------------ Node ------------------------------------
node.name: vms-002-node-1

# ---------------------------------- Paths -----------------------------------
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch

# ---------------------------------- Network -----------------------------------
network.host: 0.0.0.0
http.port: 9200

# --------------------------------- Discovery ----------------------------------
discovery.seed_hosts: ["192.168.153.132"]
cluster.initial_master_nodes: ["vms-002-node-1"]
```

On va tout d'abord il faut installer la base de données version 16 de PostgreSQL

1) Installation du **DBMS PostgreSQL**

Gestion des sources pour le **DBMS**

```
❖ sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt
  $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

```
❖ curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc|sudo
  gpg --dearmor -o /etc/apt/trusted.gpg.d/postgresql.gpg
```

```
❖ apt update -y
```

Installation des paquets pour le **DBMS**

```
❖ apt install postgresql-16
```

```
❖ apt -y install postgresql postgresql-contrib
```

Démarrage du service **PostgreSQL**

```
❖ sudo systemctl start postgresql
```

```
❖ sudo systemctl enable postgresql
```

```
psql –version : Pour vérifier la version de PostgreSQL
```

Pare-feu activé ? si oui autoriser **PostgreSQL** à le traverser

```
Iptables -I INPUT -p tcp –dport 5432 -s <ip> -j ACCEPT
```

Il existe 03 mode d'authentification sur **PostgreSQL**

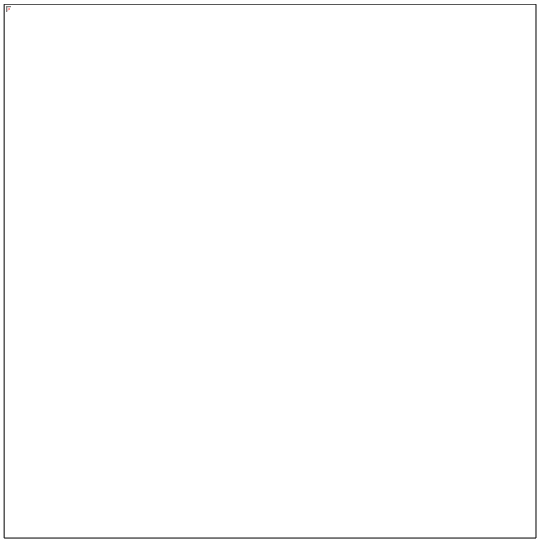
- Authentification par login/password
- Authentification de confiance

Permet à un rôle de se connecter tant que qu'une certaine condition déclarée dans **pg_hba.conf** est remplie.

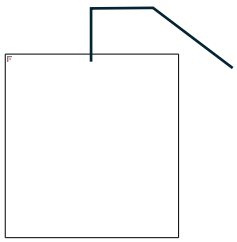
- Authentification par pairs (peer)

Idem qu'une authentification par login/password. Cependant ne peut être effectué que sur des connections locaux.

Les sources de **PostgreSQL** se trouvent dans **/etc/PostgreSQL***



Dans **/etc/postgresql-common/ :** On retrouve les sources du cluster de la bd PostgreSQL et des configurations communes entre cluster.



On va tenter de se connecter sur PostgreSQL en utilisant ces modes de connections.

On va tenter une connexion par login/password qui inclue aussi l'authentification de confiance.

- Dans le fichier **postgresql.conf,** autoriser une connexion à distance

```
data_directory = '/var/lib/post-
gresql/16/main'
hba_file = '/etc/post-
gresql/16/main/pg_hba.conf'
ident_file = '/etc/post-
gresql/16/main/pg_ident.conf'
external_pid_file = '/var/run/postgresql/16-main.pid'
listen_addresses = '*'
=
port = 5432
max_connections = 100
unix_socket_directories = '/var/run/postgresql'
ssl = on
ssl_cert_file = '/etc/ssl/certs/ssl-cert-snakeoil.pem'
ssl_key_file = '/etc/ssl/private/ssl-cert-snakeoil.key'
shared_buffers = 128MB
dynamic_shared_memory_type = posix
max_wal_size = 1GB
min_wal_size = 80MB
log_line_prefix = '%m [%p] %q%u@%d '
log_timezone = 'Etc/UTC'
cluster_name = '16/main'
datestyle = 'iso, dmy'
timezone = 'Etc/UTC'
lc_messages = 'fr_FR.UTF-8'
lc_monetary = 'fr_FR.UTF-8'
lc_numeric = 'fr_FR.UTF-8'
```

```
lc_time = 'fr_FR.UTF-8'
default_text_search_config = 'pg_catalog.french'
include_dir = 'conf.d'
```

- On doit autoriser une authentification par login/password dans le fichier **pg_hba.conf**

Pour y arriver on doit d'abord changer la méthode de chiffrement du mot de passe.

```
sed -i '/^host/s/ident/md5/' /etc/postgresql/16/main/pg_hba.conf
```

Dans notre cas **ident** n'existe pas c'est plutôt **scram-sha-256**.

```
sed -i '/^host/s/scram-sha-256/md5/' /etc/postgresql/16/main/pg_hba.conf
```

Alors que fait cette commande :

- Elle va rechercher sur l'ensemble du fichier les lignes commençant par **host**
- Substituer sans créer de fichier temporaire(tout comme nano et vim) la valeur **scram-sha-256** par **md5**

- On va aussi modifier la valeur de **peer** dans les lignes commençant par local à **trust**

```
sed -i '/^local/s/peer/trust/' /etc/postgresql/16/main/pg_hba.conf
```

- Pour terminer changer l'ipv4 en

```
# IPv4 local connections:
 host    all             all             0.0.0.0/0               md5
```

Afin de permettre l'accès à distance par toutes les adresses Ip.

- Redémarrer PostgreSQL.
- Se connecter à postgresql

```
sudo -u postgres psql
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Saisissez « help » pour l'aide.

postgres=# <ici on est dans postgres>
```

ou encore on peut décider de se connecter avec l'utilisateur postgresql

```
root@vms-002-server:~# sudo -i -u postgres
postgres@vms-002-server:~$ pwd
```

```
postgres@vms-002-server:~$ /var/lib/postgresql

postgres@vms-002-server:~$ psql <-- (commande pour entrer dans le prompte de
la base de données)
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Saisissez « help » pour l'aide.

postgres=# <Ici on est dans postgresql>
```

Conclusion : Durant la configuration de postgresql, un super utilisateur postgres est créé par default.

On peut soit se connecter directement à la BD par la première méthode soit par la seconde en utilisant .

Sécuriser le super utilisateur par default **postgres** en lui créant un mot de passe

Le mot de passe sera stocké dans le magasin de clés

```
ALTER USER postgres PASSWORD 'Str0ngP@ssw0rd';
```

```
psql -h <ip> -U postgres
```

Exemple :

```
psql -h 192.168.153.132 -U postgres
Mot de passe pour l'utilisateur postgres :
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Connexion SSL (protocole : TLSv1.3, chiffrement : TLS_AES_256_GCM_SHA384, com-
pression : désactivé)
Saisissez « help » pour l'aide.
```

Remarque :

```
root@vms-002-server:~# psql -U postgres template1
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Saisissez « help » pour l'aide.

template1=# \q
root@vms-002-server:~# psql -U postgres postgres
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Saisissez « help » pour l'aide.

postgres=#
```

template1 et postgres : sont en effet des bases de données présentent dans PostgreSQL

```
                                                    Liste des bases de données
   Nom    | Propriétaire | Encodage | Fournisseur de locale | Collationnement | Type caract. | Locale ICU | Règles
ICU : |    Droits d'accès
----------+--------------+----------+-----------------------+-----------------+--------------+------------+-------
------+-----------------------
postgres  | postgres     | UTF8     | libc                  | fr_FR.UTF-8     | fr_FR.UTF-8  |            |
      |
template0 | postgres     | UTF8     | libc                  | fr_FR.UTF-8     | fr_FR.UTF-8  |            |
      | =c/postgres          +
      |              |          |                       |                 |              |            |
      | postgres=CTc/postgres
template1 | postgres     | UTF8     | libc                  | fr_FR.UTF-8     | fr_FR.UTF-8  |            |
      | =c/postgres          +
      |              |          |                       |                 |              |            |
      | postgres=CTc/postgres
3 lignes)
```

C'est la liste des bases de données présente **\list**

- Création d'un nouvel utilisateur dans PostgreSQL

Se connecter en tant super utilisateur

- Exécuter la commande

```
CREATE ROLE lambo WITH LOGIN PASSWORD 'password$1';
```

C'est un utilisateur sans rôle de base sur la BD postgres

\du : pour avoir la liste des utilisateurs créés sur une base de données.

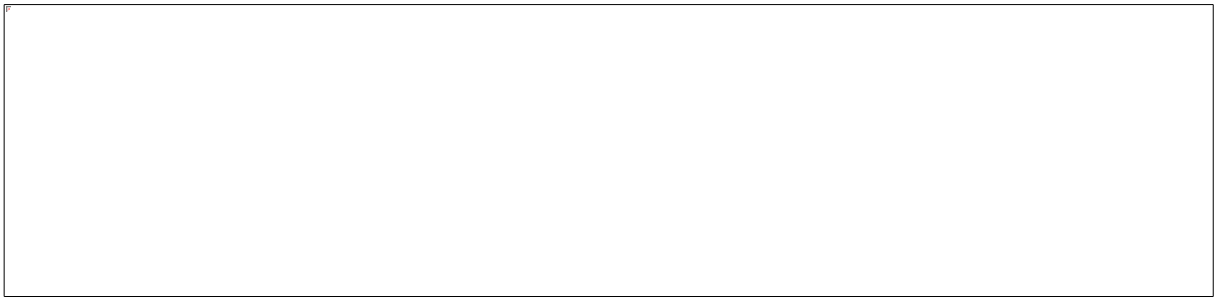
```
sudo -u postgres psql
```

Pour se connecter à la base de données :

```
psql -U postgres -d <nom de la bd>
```

Connection avec un compte que l'on connait

```
psql -U your_pg_user -h your_host -d your_database
```



On a tenté de se connecter avec l'utilisateur sudo et un utilisateur qu'on a créé sur la base de données **postgres.**

La différence est frappante dans la mesure où pour le super utilisateur :

```
Prompte : postgres=#
```

```
Prompte : postgres=>
```

Création d'une nouvelle base de données

```
CREATE DATABASE sonarqube;
```

NB : En fonction de l'utilisateur avec lequel on se connecte sur PostgreSQL, la création d'une bd est automatiquement attribuée à cet utilisateur.

On peut donner des permissions à un utilisateur sur cette BD.

On va attribuer tous les privilèges à l'utilisateur **postgres** sur notre base de données **sonarQube.**

```
GRANT ALL PRIVILEGES ON DATABASE sonarqube to postgres;
```

```
root@vms-002-server:~# psql -U postgres -d sonarqube
psql (16.3 (Ubuntu 16.3-1.pgdg22.04+1))
Saisissez « help » pour l'aide.

sonarqube=# \du
                                   Liste des rôles
 Nom du rôle |                                Attributs
-------------+-------------------------------------------------------------------------------
 lambo       |
 postgres    | Superutilisateur, Créer un rôle, Créer une base, Réplication, Contournement RLS
```

Cette commande affiche toutes les tables publiques présentes dans Postgres

```
SELECT table_name FROM information_schema.tables WHERE table_schema = 'pu-
blic';
```

C'est dans la table projects que l'on peut voir la liste de nos projets Sonarqube.

```
select * from projects ;
```

2) Configuration du server SonarQube

NB : Se rassurer qu'il existe bien sur la **Vm** le jdk java ayant la bonne version (dans notre cas pas besoin il s'agit du jdk de jenkins).

Pour arriver à cet idéal, 03 instances sont à configurer :

- Installation du Server Web Nginx : Pas nécessaire dans notre cas on a HAPROXY

- Installation de Elastic Search

On va installer la version 7.17

- Importer la clé PGP

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --
dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
```

- Ajouter les sources pour le paquet elasticSearch

```
apt install apt-transport-https
```

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg]
https://artifacts.elastic.co/packages/7.17/apt stable main" | sudo tee
/etc/apt/sources.list.d/elastic-7.17.list
```

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-
7.17.22-amd64.deb
```

SHA512

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-
7.17.22-amd64.deb.sha512
```

- On vérifie le SHA

```
shasum -a 512 -c elasticsearch-7.17.22-amd64.deb.sha512
```

- Si OK alors on install le .deb via **dpkg** (pcq Ubuntu)

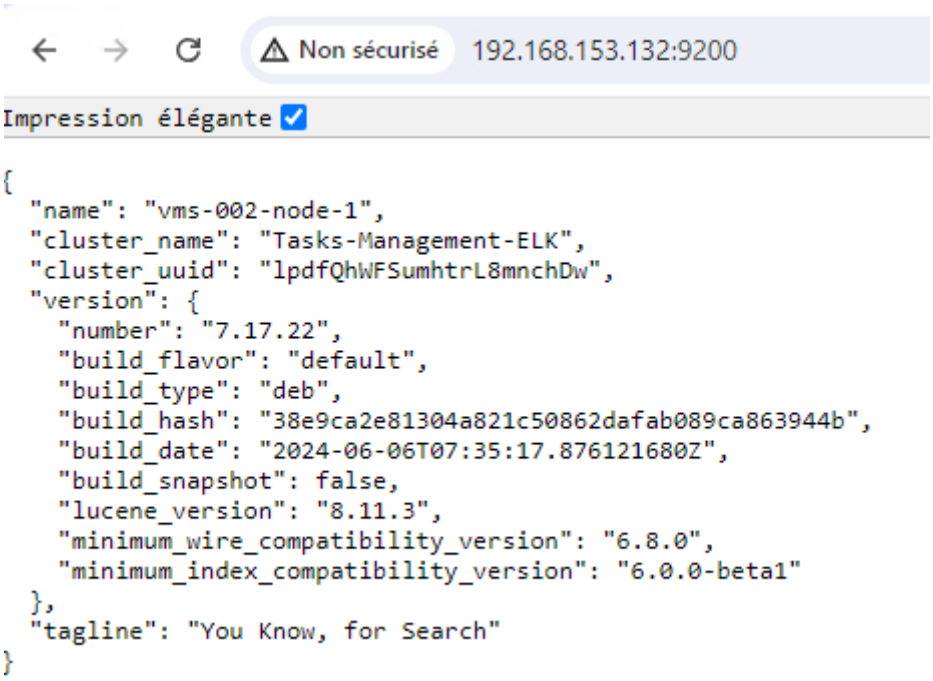
```
dpkg -i elasticsearch-7.17.22-amd64.deb
```

Dans **/etc/elasticsearch/elasticsearch.yml**

On va définir quelques valeurs

```
cluster.name: Tasks-Management-ELK
node.name: vms-002-node-1
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
#network.host: 192.168.0.1
network.host: 0.0.0.0
http.port: 9200
discovery.seed_hosts: ["192.168.153.132"]
cluster.initial_master_nodes: ["vms-002-node-1"]
```

Résultat final



- Installation du compute engine

Téléchargement de l'exécutable sonarqube

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-<version>.zip
```

```
unzip -q sonarqube-<version>.zip
```

Créer un répertoire **/opt/sonarqube**

Créer l'utilisateur et le groupe d'utilisateurs **sonarqube** n'ayant pas de sous répertoire dans **/home** ni de mot de passe.

```
adduser --system --no-create-home --group --disabled-login sonarqube
```

Rendre l'utilisateur propriétaire du répertoire **/opt/sonarqube**

```
sudo chown sonarqube:sonarqube /opt/sonarqube -R
```

Resultat

```
drwxr-xr-x 12 sonarqube sonarqube 4096 juil. 18 12:01 sonarqube
```

```
sudo mv sonarqube-9.6.1.59531 /opt/sonarqube
```

Suppression du .zip

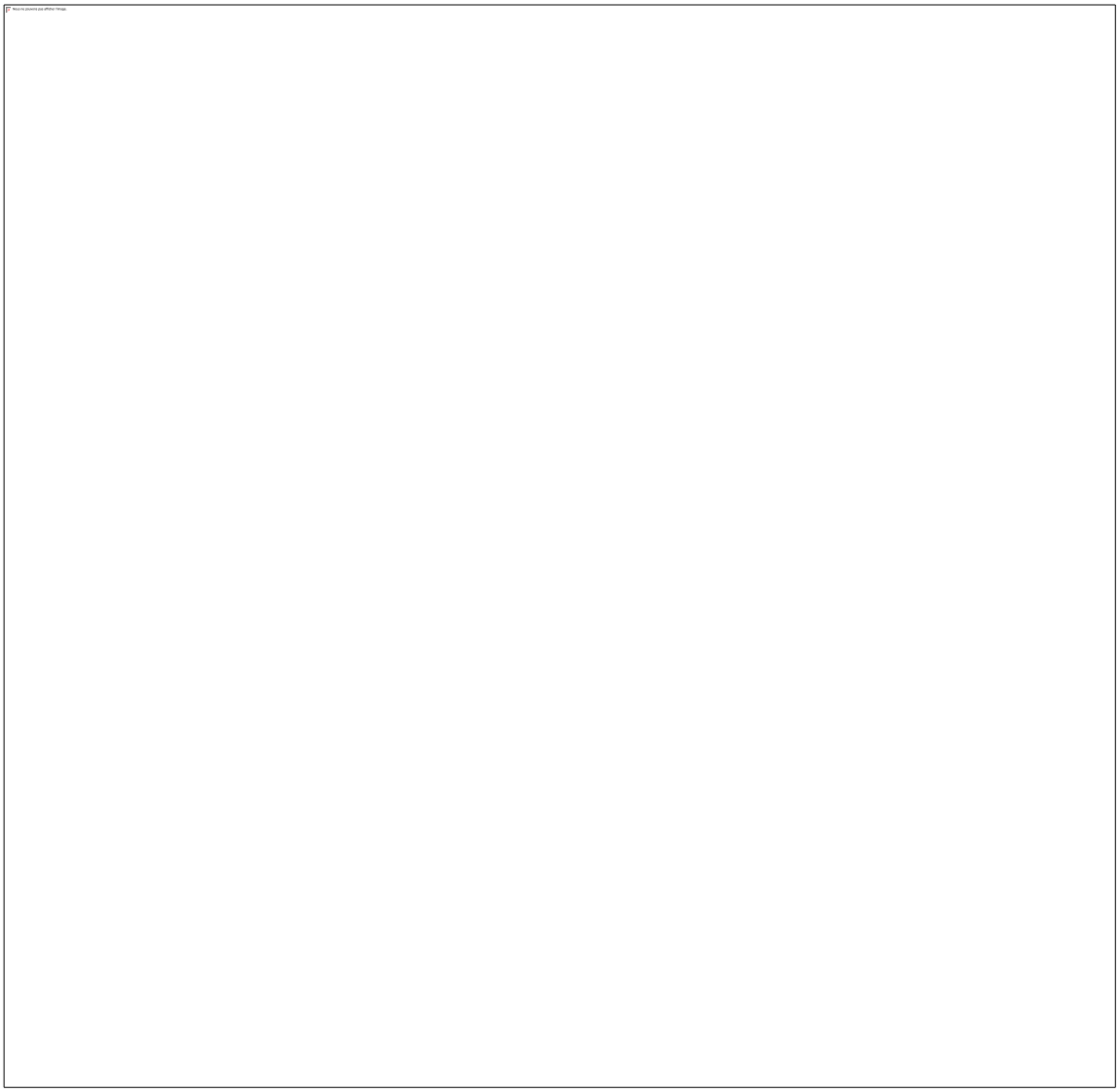
```
rm sonarqube-9.6.1.59531.zip
```

Remarque : Reproduire la configuration de ElasticSearch dans **/opt/sonarqube/elasticsearch/config/elasticsearch.yml**

Dans **/opt/sonarqube/conf/sonar.properties**

```
# DATABASE
# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC
user.
# The schema must be created first.
sonar.jdbc.username=postgres
sonar.jdbc.password=fX5uuRriG76irTAoUoJr
#----- PostgreSQL 11 or greater
# By default the schema named "public" is used. It can be overridden with the
parameter "currentSchema".
sonar.jdbc.url=jdbc:postgresql://192.168.153.132:5432/sonarqube
sonar.web.javaAdditionalOpts=-server
sonar.web.host=192.168.153.132
sonar.web.port=9000
sonar.log.level=INFO
sonar.log.level.app=INFO
sonar.log.level.web=INFO
sonar.log.level.ce=INFO
sonar.log.level.es=INFO
sonar.path.logs=logs
sonar.log.maxFiles=10
sonar.web.accessLogs.enable=true
```

Résultat final



Pas besoin d'avoir **dotnet** installé sur l'agent jenkins

Dockerignore pour choisir les fichiers qu'on ne veut pas dans le conteneur

Dockerfile doit fortement ressembler à l'environnement de dev

a) Configuration du server SonarQube pour les utilisateurs Active Directory

On souhaite que les utilisateurs Active Directory rattachés au group Administrateur de **SonarQube** puissent gérer eux-mêmes des projets.

RMQ 1 : Première chose à constater **SonarQube Community Edition** n'intègre pas **LDAP** on peut utiliser des solutions alternatives comme un RP.

RMQ 2 :

```
Community Edition: Gratuit, avec des fonctionnalités de base de SonarQube,
mais sans support LDAP.

Developer Edition: Inclut le support LDAP et d'autres fonctionnalités avan-
cées.

Enterprise Edition: Inclut toutes les fonctionnalités de la Developer Edition,
plus des fonctionnalités supplémentaires pour les grandes entreprises.

Data Center Edition: Destiné aux environnements de haute disponibilité et à
grande échelle, avec toutes les fonctionnalités des autres éditions.
```

Solution 1 : On va configurer le serveur web **Nginx** pour la gestion de l'authentification **LDAP** sur **SonarQube**.

- Installation et configuration de Nginx

```
sudo apt install libnginx-mod-http-auth-ldap
```

C'est un module **ldap** pour nginx. Suivre juste un tutoriel pour la configuration LDAP sous Nginx.

- **HAproxy** comme RP de **Nginx**

Configurer le front et le back HAproxy pour tomber sur le serveur web Nginx.

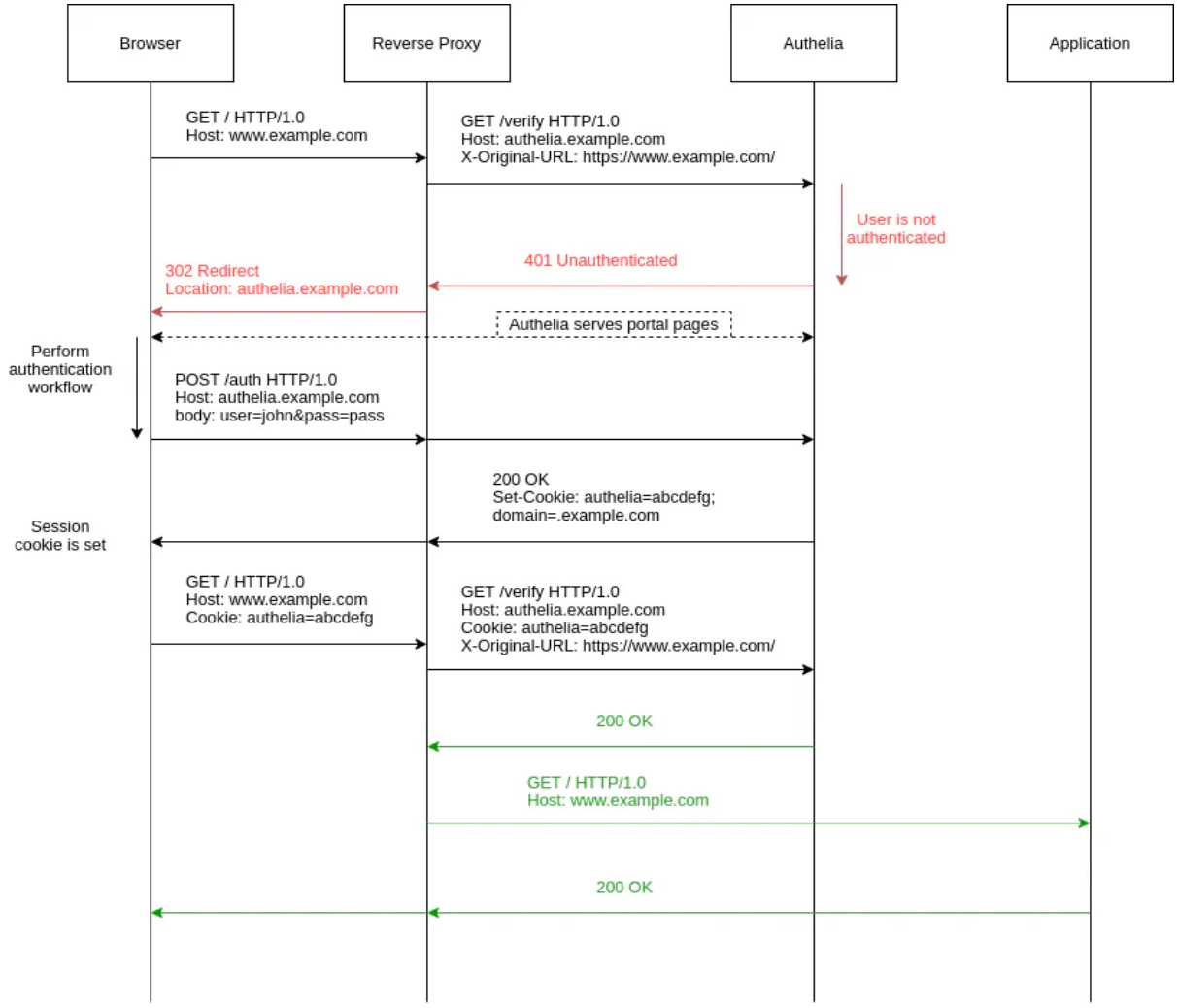
Solution 2 :

On va installer un service d'authentification intermédiaire conçu pour **HAproxy** et LDAP.

On en distingue deux sortes : **Authelia** et **oauth2_proxy.**

- Installation d'un service d'authentification **LDAP** et intégration de sonarqube

Architeture du service Authelia



b) Gestion du server SonarQube comment interpréter les problèmes

Références

https://computingforgeeks.com/install-and-configure-postgresql-on-ubuntu/#4-5-getting-started-with-postgresql | Configuration du DBMS PostgreSQL

https://docs.vultr.com/how-to-use-sonarqube-on-ubuntu-22-04-lts | Installation SonarQube sur linux Ubuntu

https://docs.sonarsource.com/sonarqube/latest/setup-and-upgrade/install-the-server/introduction/ | SonnarQube Doc Officiel 10.6

https://www.elastic.co/guide/en/elasticsearch/reference/7.17/deb.html | Installation ElasticSearch

https://www.authelia.com/configuration | Pour le service intermédiaire permettant de rallier HAproxy à LDAP pour une authentification AD.

	Services	Dernières versions	Versions installées		
	Jenkins	2.467	2.441	Plugins	Versions
				AD plugins	
				Docker plugins	