🚀 INSTALLATION ET CONFIGURATION DE GITLAB SERVER

🚀 1. Création de la Machine Virtuelle GitLab avec Vagrant

📦 Objectif

Créer une VM Ubuntu 22.04 avec VirtualBox pour héberger GitLab Community Edition.

🗂️ Configuration du Vagrantfile

```
GITLAB_NODE = { :hostname ⇒ "gitlab-server", :ip ⇒ "192.168.193.130" }

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/jammy64"  # Image de base Ubuntu 22.04
  config.vm.boot_timeout = 600

  config.vm.define GITLAB_NODE[:hostname] do |node_config|
    node_config.vm.hostname = GITLAB_NODE[:hostname]  # Nom de la VM
    node_config.vm.network "private_network", ip: GITLAB_NODE[:ip]  # IP fixe

    node_config.vm.provider "virtualbox" do |vb|
      vb.name = GITLAB_NODE[:hostname]  # Nom dans VirtualBox
      vb.memory = 6144  # 6 Go RAM pour GitLab
      vb.cpus = 3       # 3 CPU
      vb.gui = false    # Pas besoin d'interface graphique
    end

    node_config.vm.provision "shell", inline: <<-SHELL
```

```
    apt update && apt upgrade -y
    apt install -y curl openssh-server ca-certificates tzdata perl
  SHELL
 end
end
```

🧪 Lancer la VM étape par étape

```
# Créer le dossier de travail
mkdir gitlab-server
cd gitlab-server

# Initialiser le fichier Vagrantfile avec le contenu ci-dessus
# Puis lancer la VM :
vagrant up

# Se connecter à la VM
vagrant ssh
```

🛠️ 2. Installation de GitLab CE sur Ubuntu 22.04

🔧 Mise à jour du système & configuration du pare-feu

```
sudo apt update && sudo apt upgrade -y  # Mise à jour du système
sudo apt install -y ufw                 # Installer UFW (pare-feu)
sudo ufw allow http
sudo ufw allow https
sudo ufw allow OpenSSH
sudo ufw reload
sudo ufw enable
```

🔧 Installer GitLab CE

```
# Installer les dépendances nécessaires
sudo apt install -y curl openssh-server ca-certificates tzdata perl

# Ajouter le dépôt GitLab
curl -fsSL https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/scri
pt.deb.sh | sudo bash

# Installer GitLab Community Edition
sudo apt install -y gitlab-ce
```

🔧 Configurer l'URL d'accès à GitLab

```
sudo nano /etc/gitlab/gitlab.rb
# Modifier la ligne :
external_url 'http://192.168.193.130'
# Enregistrer avec Ctrl+O, puis quitter avec Ctrl+X
```

🔃 Appliquer la configuration GitLab

```
sudo gitlab-ctl reconfigure
sudo gitlab-ctl status  # Vérifier que tous les services sont OK
```

🌐 Accès via navigateur

> Ouvre ton navigateur et va sur : http://192.168.193.130

- Identifiant : `root`
- Mot de passe initial :

```
sudo cat /etc/gitlab/initial_root_password
```

🔐 3. Clés SSH & Tokens GitLab

🔑 Générer une clé SSH sur ta VM ou machine

```
ssh-keygen -t rsa -b 4096 -C "ton-email@example.com"
# Appuie sur Entrée jusqu'à la fin pour accepter les chemins par défaut
```

🔑 Ajouter la clé SSH à l'agent

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa
```

🔑 Copier la clé publique

```
cat ~/.ssh/id_rsa.pub
```

➡️ Va sur GitLab → **User Settings → SSH Keys → Add Key**

🧪 Tester la connexion SSH

```
ssh -T git@192.168.193.130
```

Tu devrais voir : `Welcome to GitLab, @ton-utilisateur`

🎟️ Créer un Token d'accès personnel

- Dans GitLab → User Settings → Access Tokens
- Nom : `Joget-Access`
- Permissions : ✅ `api`, `read_repository`, `write_repository`
- Clique sur **Create token**, et copie-le !

🔗 4. Intégrer GitHub à GitLab via OAuth (optionnel)

1. Sur GitHub : Créer une App OAuth

- Developer Settings → OAuth Apps → New OAuth App

- Homepage URL : `http://192.168.193.130`
- Callback URL : `http://192.168.193.130/users/auth/github/callback`

2. Sur GitLab : Modifier `/etc/gitlab/gitlab.rb`

```
external_url 'http://192.168.193.130'

# Activer l'auth GitHub
gitlab_rails['omniauth_enabled'] = true
gitlab_rails['omniauth_allow_single_sign_on'] = ['github']
# gitlab_rails['omniauth_auto_sign_in_with_provider'] = 'github'  # commentée pour afficher l'écran de login

gitlab_rails['omniauth_providers'] = [
 {
  "name" ⇒ "github",
  "app_id" ⇒ "TON_CLIENT_ID",
  "app_secret" ⇒ "TON_CLIENT_SECRET",
  "args" ⇒ { "scope" ⇒ "user:email" }
 }
]
```

```
sudo gitlab-ctl reconfigure
```

3. Désactiver validation manuelle

```
gitlab_rails['gitlab_signup_enabled'] = true
```

🛡️ 5. Renforcer la sécurité de GitLab

Editer `/etc/gitlab/gitlab.rb` pour ajouter :

```
# Protection bruteforce
gitlab_rails['rack_attack_git_basic_auth'] = {
  'enabled' => true
}

# Politique de mot de passe fort
gitlab_rails['gitlab_password_length'] = {
  'minimum' => 12,
  'maximum' => 128
}
```

```
sudo gitlab-ctl reconfigure
```

👬 6. Gestion des utilisateurs et groupes (RBAC)

1. Créer un groupe `melkart`

- Interface GitLab → `Groups → New Group`
- Nom : melkart → URL : http://192.168.193.130/groups/melkart
- Visibilité : Private → Créer

2. Ajouter des projets/sous-groupes

- `melkart/joget-app`
- `melkart/infra-ansible`, etc.

3. Ajouter des membres avec rôles

- Aller dans : `melkart → Members`
- Invite : `motrabelsi10` → Choisir rôle : Maintainer / Developer

Récapitulatif des rôles :

Rôle	Capacité

Guest	Lecture seule
Reporter	Clonage, téléchargement
Developer	Push, merge, CI/CD
Maintainer	Gère branches protégées, settings, pipelines
Owner	Gère tout (groupes uniquement, pas pour projets individuels)

📂 7. Intégrer GitLab avec Joget (Git Integration)

A. Paramètres Git dans Joget

Aller dans `Settings → Git Integration`

Champ Git	Valeur
Repository URL	`http://192.168.193.130/melkat-joget/joget-app.git`
Username	`motrabelsi10`
Token (comme password)	`glpat-xxxxxxx...`
Always Pull	✅
Auto Sync DB from Git	✅

💾 Clique sur Save → puis "Test Connection"

B. Organisation des branches

Branche	Utilisation
main	Production stable
app_1	Pré-production / recette
app_2	Feature d'un dev spécifique
app_3+	Autres branches de test ou expérimentations

C. Exemple de workflow depuis Joget :

```
cd C:/Joget-DX8/wflow/app_src/app/app_1

# Initialisation Git
```

```
git init

git remote add origin http://192.168.193.130/melkat-joget/joget-app.git

# Créer une branche recette
git checkout -b app_1

# Ajouter les fichiers
git add .
git commit -m "Initial commit Joget app (app_1)"

# Pousser la branche vers GitLab
git push -u origin app_1
```

D. Depuis Joget :

1. Dans `Settings → Git Integration` → changer `Git Branch` → `app_1`
2. Clique sur "Pull"
3. Travaille sur l'application
4. Clique sur "Push" pour sauvegarder

🔁 8. Bonnes pratiques de collaboration

- **Chaque développeur** travaille sur une branche dédiée (ex: `app_2`, `app_3`)
- Les changements sont intégrés dans `main` via **Merge Requests GitLab**

1. Télécharge Vault depuis HashiCorp

wget https://releases.hashicorp.com/vault/1.14.4/vault_1.14.4_linux_amd64.zip

2. Dézippe le binaire

unzip vault_1.14.4_linux_amd64.zip

3. Déplace Vault dans le PATH

sudo mv vault /usr/local/bin/

4. Vérifie l’installation

vault --version