1) Pré-requis côté machine hôte

- Linux avec Docker Engine récent et Compose v2 (CLI docker compose).
- Ports libres : **8080** (UI Jenkins) et **50000** (agents JNLP, optionnel).

2) Créer l'arborescence Jenkins

sudo mkdir -p /var/jenkins_home/{compose,cache}

sudo chmod 777 /var/jenkins_home

cd /var/jenkins_home/compose

docker network create jenkins-net

On va construire **une image Jenkins custom** (basée sur jenkins/jenkins:lts-jdk21) avec le **CLI Docker + docker compose plugin** à l'intérieur (pour que tes pipelines puissent faire docker build, docker run, etc.).

Dockerfile

```
# /opt/jenkins/compose/Dockerfile
FROM jenkins/jenkins:lts-jdk21
# On installe docker-ce-cli + docker compose plugin + git + curl
USER root
RUN apt-get update && apt-get install -y --no-install-recommends \
      ca-certificates curl gnupg git \
  && install -m 0755 -d /etc/apt/keyrings \
  && curl -fsSL https://download.docker.com/linux/debian/gpg \
       | gpg --dearmor -o /etc/apt/keyrings/docker.gpg \
  && echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] \
      https://download.docker.com/linux/debian bookworm stable" \
      > /etc/apt/sources.list.d/docker.list \
  && apt-get update && apt-get install -y --no-install-recommends \
      docker-ce-cli docker-compose-plugin \
  && rm -rf /var/lib/apt/lists/*

# Plugins utiles démarrage (optionnel)
RUN jenkins-plugin-cli --plugins \
  "workflow-aggregator docker-workflow credentials-binding github-branch-
source" " \
  "blueocean docker-workflow json-path-api"
```

Pourquoi USER root ? C'est la méthode la plus simple pour éviter les problèmes de droits sur /var/run/docker.sock.

3) docker-compose : lancer Jenkins avec accès au Docker du host

docker-compose.yml

```
# /var/jenkins_home/compose/docker-compose.yml
services:
  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: jenkins
    restart: unless-stopped
    ports:
      - "8080:8080"    # UI Jenkins
      - "50000:50000"  # agents JNLP (optionnel)
    environment:
      - JAVA_OPTS=-Djenkins.install.runSetupWizard=true
      - JENKINS_OPTS=--prefix=/jenkins
      - DOCKER_HOST=unix:///var/run/docker.sock
    volumes:
      - /var/jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    networks:
      - jenkins-net
networks:
  jenkins-net:
    external: true Démarrage :
```

cd :/var/jenkins_home/compose

sudo docker compose build

sudo docker compose up -d

4) Récupérer le mot de passe admin initial

sudo docker exec -it jenkins bash -lc 'cat /var/jenkins_home/secrets/initialAdminPassword'

- Ouvre **http://<IP_ou_hostname>:8080/jenkins**
- Colle le mot de passe.
- Installe les plugins recommandés.
- Crée ton utilisateur admin.

5) Vérifier l'accès Docker depuis Jenkins

Dans **Manage Jenkins → Script Console**, lance :

println "whoami: " + "id".execute().text

println "docker version:\n" + "docker version".execute().text

Ou crée un job "Freestyle" avec un build step **Execute shell** :

Étapes — Crée un nouveau job

1. Clique sur **"New Item"** (ou "Nouveau Item").
2. Donne un nom à ton job, par exemple :
   docker-access-test
3. Sélectionne **"Freestyle project"**.
4. Clique sur **OK**.

Configure le build step

1. Dans la page de configuration du job, fais défiler jusqu'à **Build** (ou "Construction").
2. Clique sur **Add build step → Execute shell**.
3. Colle le script suivant :

```
echo "=== Test Docker depuis Jenkins ==="
docker version
echo
echo "=== Lancer un conteneur test ==="
docker run --rm hello-world
echo
echo "=== Liste des conteneurs en cours ==="
docker ps -a
echo "=== Test docker compose ==="
docker compose version
```

4. Clique sur **Save**

🧩 **Étape 1 — Crée ton token Git**

🔹 **Si tu utilises GitHub**

1. Va sur ton profil GitHub → **Settings → Developer Settings → Personal access tokens → Tokens (classic)**.
2. Clique **Generate new token → classic**.
3. Coche :
    - repo (accès complet aux dépôts privés)
    - read:org (si Jenkins doit lire les branches d'une organisation)
4. Copie ton token et garde-le en lieu sûr (tu ne le reverras plus).

Étape 2 — Crée le credential dans Jenkins

1. Ouvre ton Jenkins : http://<ip>:8080/jenkins
2. Clique **Manage Jenkins → Credentials → System → Global credentials (unrestricted)** (ou via "Manage Credentials" → "(global)")
3. Clique **Add Credentials**.
4. Remplis le formulaire :
    - **Kind** → Username with password
    - **Username** → ton identifiant GitHub/GitLab (ex: eliesclevory)
    - **Password** → colle le **token personnel**
    - **ID** → git-creds
    - **Description** → Access token for private Git repository
5. Clique **Create**.

👉 Jenkins stocke ce secret chiffré.

Objectif : un **Nexus Dockerisé** qui héberge à la fois :

- 🐬 **les images Docker** (registry privé pour Jenkins),
- 📦 **les artefacts** (fichiers .jar, .zip, .tgz, etc. si tu veux en stocker).

🧩 1. Dossier & structure

Sur ton hôte (exemple /opt/nexus)

sudo mkdir -p /opt/nexus/data

sudo chown -R 200:200 /opt/nexus/data

vim /var/jenkins_home/compose/docker-compose.yml

Ajouter ceci

```
  nexus:
    image: sonatype/nexus3:3.68.1
    container_name: nexus
    restart: unless-stopped
    user: "200:200"
    ports:
      - "8081:8081"   # Interface Web Nexus
      - "8880:8880"   # Docker registry local
    environment:
      - INSTALL4J_ADD_VM_PARAMS=-Xms512m -Xmx2048m
    volumes:
      - /opt/nexus/data:/nexus-data
    networks:
      - jenkins-net
```

cd /var/jenkins_home/compose/

docker compose down

docker compose up -d

Accès

Service	URL	Description
Jenkins	http://<IP>:8080	CI/CD principal
Nexus	http://<IP>:8081	Interface Repository
Docker Registry	http://<IP>:8880	Registry privé

🔐 Récupération du mot de passe admin Nexus

docker exec nexus cat /nexus-data/admin.password

Copie le mot de passe affiché et colle-le dans l'interface.
👉 Clique sur **"Sign in"**.

Setup wizard

Définis un nouveau mot de passe admin

Choisis ton mot de passe personnel (par ex. password).

"Enable anonymous access?"

Tu peux choisir :

- **Keep anonymous access enabled** ✅ (recommandé en dev)
  Cela te permet de **télécharger des artefacts sans login**.
- ou **Disable** si tu veux que tout soit privé.

Clique sur **Next**.

Fin du setup

Tu arriveras sur la page principale "Welcome to Nexus Repository Manager".

🧱 **Création du Docker Registry privé**

🔹 6️⃣ **Va dans le menu latéral :**

Settings → Repositories → Create repository

🔹 7️⃣ **Choisis le format :**

👉 Clique sur **docker (hosted)**

🔹 8️⃣ **Configure le dépôt**

Champ	Valeur
Name	local-docker
HTTP	✅ activé
HTTP Port	8880
Deployment policy	**Allow redeploy**
Blob store	default (laisser par défaut)
Enable Docker V1 API	❌ (désactivé, inutile)
Allow anonymous docker pull	✅ (pour dev interne, sinon décocher)

echo "192.168.1.30 local-registry.lab.local local-registry" >> /etc/hosts

vas sur :

http://local-registry:8880/v2/_catalog

Tu devrais obtenir :

{"repositories":[]}

🧩 1️⃣ Autoriser un registre HTTP non sécurisé

Par défaut, Docker refuse tout http:// registry.
Tu dois donc le déclarer comme “insecure registry”.

Sur la machine hôte (et tous les hôtes Docker qui pousseront vers Nexus) :

Modifie ou crée le fichier :

sudo nano /etc/docker/daemon.json

```
{
  "insecure-registries": [
    "local-registry.lab.local:8880",
    "local-registry:8880"
  ],
  "live-restore": true
}
```

systemctl daemon-reexec

systemctl restart docker

docker info | grep -A4 "Insecure Registries"

Insecure Registries:

local-registry.lab.local:8880

local-registry:8880

::1/128

127.0.0.0/8

Dans Nexus -> settings -> roles

Crée un rôle nx-docker

Applied privileges : nx-repository-admin-docker-local-registry-*

Dans Nexus -> settings -> users
crée un user : elies avec mot de passe : password et Roles : nx-docker(granted)

✅ **Test rapide**

Depuis ta machine hôte :

docker login http://local-registry:8880

Username: elies

Password: ton_mot_de_passe_admin

⚙️ 1️⃣ **Ce que fait SonarQube**

SonarQube, c'est ton **moteur d'analyse de qualité de code**.
Tu pourras :

- 🚨 Détecter bugs, code smells, vulnérabilités et duplications.
- 🧪 Vérifier le taux de couverture des tests (reprend ton lcov.info Jest).
- 🧠 Définir des "Quality Gates" : par ex. build fail si couverture < 80 %.
- 📊 Avoir un tableau de bord complet par projet, accessible via web.
- 🔗 Coupler Jenkins, Nexus et même GitHub/GitLab.

🧩 2️⃣ **Étape 1 – Ajouter SonarQube à ton docker-compose.yml**

👉 Tu vas ajouter un service sonarqube **sur le même réseau** que Jenkins et Nexus.

```
 sonarqube:
  image: sonarqube:10.5-community
  container_name: sonarqube
  restart: unless-stopped
  environment:
   - SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true
  ports:
   - "9000:9000"
  volumes:
   - /opt/sonarqube/data:/opt/sonarqube/data
   - /opt/sonarqube/extensions:/opt/sonarqube/extensions
   - /opt/sonarqube/logs:/opt/sonarqube/logs
  networks:
   - jenkins-net
```

Arborescence sur l'hôte (exemple)

Tu peux créer les dossiers une fois pour toutes :

sudo mkdir -p /opt/sonarqube/{data,extensions,logs}

sudo chown -R 999:999 /opt/sonarqube

chmod -R 755 /opt/sonarqube

🧠 UID 999 = utilisateur sonarqube à l'intérieur du conteneur.
Si tu ne fais pas le chown, SonarQube risque de ne pas démarrer à cause d'un Permission denied.

🔮 **Contenu de chaque dossier**

Dossier	**Contenu**	**Persistance**
/opt/sonarqube/data	Indexes Elasticsearch, DB intégrée	**Obligatoire**
/opt/sonarqube/extensions	Plugins Sonar (ajoutés manuellement)	**Obligatoire**
/opt/sonarqube/logs	Logs applicatifs (web, es, ce)	Optionnel mais recommandé

Sonar sera accessible sur http://sonarqube:9000
ou depuis ton hôte : http://<ip_serveur>:9000

🧠 3️⃣ Étape 2 – Configuration initiale SonarQube

Une fois lancé :

1. Va sur http://localhost:9000
2. Identifie-toi :
   Login : admin
   Password : admin
3. Change le mot de passe (obligatoire)
4. Va dans : **Administration → Security → Users → Tokens**
5. Crée un token (nom : jenkins-token)
   → Copie-le, on l'utilisera dans Jenkins.