Guacamole Docker - Configuration Complète

Une image Docker personnalisée de Apache Guacamole avec support complet pour PostgreSQL, LDAP, OpenID Connect et de nombreuses options de configuration.

📋 Fonctionnalités

- ✅ **Apache Guacamole 1.5.4** avec toutes les extensions
- ✅ **PostgreSQL** comme base de données backend
- ✅ **LDAP** pour l'authentification d'entreprise
- ✅ **OpenID Connect** pour SSO (Single Sign-On)
- ✅ **Mode Debug** configurable via variables d'environnement
- ✅ **JMX** pour le monitoring et la supervision
- ✅ **Configuration JVM** complète et optimisée
- ✅ **Logs rotatifs** avec niveaux configurables
- ✅ **Support Prometheus/Grafana** pour métriques (optionnel)

🚀 Démarrage Rapide

1. Cloner ou créer la structure

```bash
mkdir guacamole-docker
cd guacamole-docker

# Créer les répertoires nécessaires
mkdir -p init extensions prometheus grafana/provisioning
```

2. Placer les fichiers

Copier tous les fichiers fournis :

- `Dockerfile`
- `docker-entrypoint.sh`
- `docker-compose.yml`
- `.env.example`
- `init/01-init-guacamole-db.sql`

3. Configuration

```bash
# Copier le fichier d'environnement
cp .env.example .env

# Éditer les variables selon vos besoins
nano .env
```

4. Démarrage

```bash
# Build et démarrage des services
docker-compose up -d

# Vérifier les logs
docker-compose logs -f guacamole
```

5. Accès

- **Guacamole**: http://localhost:8080/guacamole
  - Login par défaut: `guacadmin` / `guacadmin`
- **Grafana** (si activé): http://localhost:3000
  - Login: `admin` / `admin`

📊 Variables d'Environnement Principales

Configuration de Base

Variable	Description	Défaut
`POSTGRES_DATABASE`	Nom de la base de données	`guacamole_db`
`POSTGRES_USER`	Utilisateur PostgreSQL	`guacamole`
`POSTGRES_PASSWORD`	Mot de passe PostgreSQL	⚠️ À définir

Configuration des Logs

Variable	Description	Défaut
`LOG_LEVEL`	Niveau de log (debug, info, warn, error)	`info`
`ENABLE_DEBUG`	Active le mode debug complet	`false`
`LOG_FILE`	Chemin du fichier de log	`/var/log/guacamole/guacamole.log`
`LOG_MAX_FILE_SIZE`	Taille max des logs	`10MB`
`LOG_MAX_HISTORY`	Nombre de fichiers à conserver	`7`

Configuration JVM

Variable	Description	Défaut
`JVM_HEAP_MIN`	Heap minimum	`256m`
`JVM_HEAP_MAX`	Heap maximum	`1g`
`JVM_METASPACE_SIZE`	Taille Metaspace	`128m`
`ENABLE_JMX`	Active JMX pour monitoring	`false`
`JMX_PORT`	Port JMX	`9090`

Configuration LDAP

Variable	Description	Défaut
`LDAP_HOSTNAME`	Serveur LDAP	`` (désactivé)
`LDAP_PORT`	Port LDAP	`389`
`LDAP_USER_BASE_DN`	Base DN des utilisateurs	``
`LDAP_SEARCH_BIND_DN`	DN pour recherche	``
`LDAP_ENCRYPTION_METHOD`	Méthode de chiffrement	`none`

Configuration OpenID

Variable	Description	Défaut
`OPENID_ISSUER`	Issuer OpenID	`` (désactivé)
`OPENID_CLIENT_ID`	Client ID	``
`OPENID_CLIENT_SECRET`	Client Secret	``
`OPENID_REDIRECT_URI`	URI de redirection	``

🔧 Configurations Avancées

Mode Debug

Pour activer le mode debug complet :

```bash
# Dans .env
ENABLE_DEBUG=true
LOG_LEVEL=debug
```

Cela activera :

- Logs détaillés au niveau DEBUG
- Port de debug Java (5005)
- Logs GC (Garbage Collector)
- Traces détaillées des requêtes SQL

Monitoring avec JMX

```bash
# Dans .env
ENABLE_JMX=true
JMX_PORT=9090
JMX_AUTHENTICATE=true
JMX_USERNAME=monitoring
JMX_PASSWORD=SecurePassword
```

Connexion avec JConsole ou VisualVM :

```bash
jconsole localhost:9090
```

Monitoring avec Prometheus/Grafana

```bash
# Démarrer avec le profil monitoring
docker-compose --profile monitoring up -d

# Accès Grafana: http://localhost:3000
```

Configuration LDAP Active Directory

Exemple pour Active Directory :

```bash
LDAP_HOSTNAME=dc.example.com
LDAP_PORT=389
LDAP_ENCRYPTION_METHOD=starttls
LDAP_USER_BASE_DN=CN=Users,DC=example,DC=com
LDAP_USERNAME_ATTRIBUTE=sAMAccountName
LDAP_SEARCH_BIND_DN=CN=svc_guacamole,CN=Users,DC=example,DC=com
LDAP_SEARCH_BIND_PASSWORD=ServiceAccountPassword
LDAP_USER_SEARCH_FILTER=(&(objectClass=user)(sAMAccountName={username}))
```

Configuration OpenID avec Keycloak

```bash
OPENID_ISSUER=https://keycloak.example.com/realms/myrealm
OPENID_AUTHORIZATION_ENDPOINT=https://keycloak.example.com/realms/myrealm/protocol/openid-connect/a
OPENID_TOKEN_ENDPOINT=https://keycloak.example.com/realms/myrealm/protocol/openid-connect/token
OPENID_JWKS_ENDPOINT=https://keycloak.example.com/realms/myrealm/protocol/openid-connect/certs
OPENID_CLIENT_ID=guacamole
OPENID_CLIENT_SECRET=client-secret-here
OPENID_REDIRECT_URI=https://guacamole.example.com/guacamole/
```

🔍 Optimisation des Performances

Pour environnement de production

```bash
# JVM
JVM_HEAP_MIN=1g
JVM_HEAP_MAX=4g
JVM_METASPACE_SIZE=256m
JVM_MAX_METASPACE_SIZE=512m

# Tomcat
TOMCAT_MAX_THREADS=400
TOMCAT_MIN_SPARE_THREADS=25
TOMCAT_MAX_CONNECTIONS=20000

# PostgreSQL
POSTGRES_MAX_CONNECTIONS=50
POSTGRES_CONNECTION_TIMEOUT=60

# Cache
CACHE_ENABLED=true
CACHE_MAX_SIZE=50000
CACHE_TTL=1800
```

Pour beaucoup d'utilisateurs simultanés

```bash
# Augmenter les connexions
TOMCAT_MAX_THREADS=800
TOMCAT_MAX_CONNECTIONS=50000
POSTGRES_MAX_CONNECTIONS=100

# Augmenter la mémoire
JVM_HEAP_MAX=8g
JVM_DIRECT_MEMORY_SIZE=2g
```

📝 Commandes Utiles

Logs

```bash
# Voir tous les logs
docker-compose logs -f

# Logs d'un service spécifique
docker-compose logs -f guacamole

# Logs avec timestamps
docker-compose logs -t -f guacamole

# Dernières 100 lignes
docker-compose logs --tail=100 guacamole
```

Maintenance

```bash
# Redémarrer un service
docker-compose restart guacamole

# Reconstruire l'image
docker-compose build --no-cache guacamole

# Backup de la base de données
docker exec guacamole-postgres pg_dump -U guacamole guacamole_db > backup.sql

# Restauration
docker exec -i guacamole-postgres psql -U guacamole guacamole_db < backup.sql
```

Debug

```bash
# Accéder au container
docker exec -it guacamole-web bash

# Vérifier la configuration
docker exec guacamole-web cat /etc/guacamole/guacamole.properties

# Voir les processus Java
docker exec guacamole-web ps aux | grep java

# Statistiques mémoire
docker stats guacamole-web
```

🔒 Sécurité

Recommandations

1. **Changer les mots de passe par défaut**
   - Mot de passe administrateur Guacamole
   - Mot de passe PostgreSQL
   - Mots de passe JMX si activé
2. **Utiliser HTTPS**

```yaml
# Ajouter un reverse proxy nginx
nginx:
 image: nginx:alpine
 volumes:
  - ./nginx.conf:/etc/nginx/nginx.conf
  - ./certs:/etc/nginx/certs
 ports:
  - "443:443"
```

3. **Restreindre les accès réseau**
   - Ne pas exposer PostgreSQL publiquement
   - Utiliser des réseaux Docker isolés
   - Configurer un firewall
4. **Activer l'authentification 2FA**

```bash
ENABLE_TOTP=true
```

🐛 Dépannage

Problème de connexion à PostgreSQL

```bash
# Vérifier que PostgreSQL est démarré
docker-compose ps postgres

# Vérifier les logs
docker-compose logs postgres

# Tester la connexion
docker exec -it guacamole-postgres psql -U guacamole -d guacamole_db
```

Extensions non chargées

```bash
# Vérifier la présence des JAR
docker exec guacamole-web ls -la /etc/guacamole/extensions/

# Vérifier les logs pour erreurs de chargement
docker-compose logs guacamole | grep -i extension
```

Erreur OutOfMemory

```bash
# Augmenter la heap JVM
JVM_HEAP_MAX=2g

# Redémarrer
docker-compose restart guacamole
```

📚 Documentation

- Documentation officielle Guacamole
- Configuration LDAP
- Configuration OpenID
- API REST Guacamole

📄 Licence

Apache License 2.0 (même licence que Apache Guacamole)

🤝 Support

Pour toute question ou problème :

1. Vérifier la documentation officielle
2. Consulter les logs avec `docker-compose logs`
3. Activer le mode debug pour plus de détails