

Tutoriel LiveKit Self-Hosted avec Docker

Prérequis

- Docker et Docker Compose installés
- 4+ Go de RAM recommandé
- Pour la production : domaine et certificats SSL

1. Installation en Local

1.1 Configuration de base

`docker-compose.local.yml` :

```
yaml
version: '3.8'

services:
  livekit:
    image: livekit/livekit:latest
    container_name: livekit
    restart: unless-stopped
    ports:
      - "7880:7880" # WebRTC ports
      - "7881:7881" # HTTP/WebSocket
      - "7882:7882" # gRPC (optionnel)
    environment:
      - LIVEKIT_KEYS_API_KEY=devkey
      - LIVEKIT_KEYS_API_SECRET=devsecret
      - LIVEKIT_KEYS_SET=1
      - LIVEKIT_NODE_IP=127.0.0.1
      - LIVEKIT_RTC_TCP_PORT=7880
      - LIVEKIT_RTC_UDP_PORT=7880
      - LIVEKIT_PORT=7881
    volumes:
      - ./data:/data
    command: sh -c "livekit-server --dev --bind 0.0.0.0"

# Redis (optionnel pour développement)
```

```
redis:
  image: redis:7-alpine
  container_name: livekit-redis
  restart: unless-stopped
  ports:
    - "6379:6379"
```

1.2 Configuration avancée avec Redis

Créez `livekit.yaml`:

```
yaml
port: 7881
rtc:
  tcp_port: 7880
  udp_port: 7880
  port_range_start: 0
  port_range_end: 0
keys:
  APIInbB7QvDE9h6dZ: nqZ6zHJJk8omZ0KIxhaZ4DpMuG4sx7t0cF9kMEXz9MU
redis:
  address: redis:6379
  username: ""
  password: ""
  db: 0
logging:
  json: false
  level: debug
```

`docker-compose.local-redis.yml`:

```
yaml
version: '3.8'

services:
  livekit:
    image: livekit/livekit:latest
    container_name: livekit
    restart: unless-stopped
    ports:
      - "7880:7880"
      - "7881:7881"
      - "7882:7882"
```

```

volumes:
  - ./livekit.yaml:/livekit.yaml
command: livekit-server --config /livekit.yaml --dev
depends_on:
  - redis

redis:
  image: redis:7-alpine
  container_name: livekit-redis
  restart: unless-stopped
  command: redis-server --appendonly yes
  volumes:
    - ./redis-data:/data

```

1.3 Démarrer en local

```

bash

# Mode simple
docker-compose -f docker-compose.local.yml up -d

# Mode avec Redis
docker-compose -f docker-compose.local-redis.yml up -d

# Vérifier les logs
docker logs -f livekit

Accédez à : http://localhost:7881

```

2. Configuration pour la Production

2.1 Fichier de configuration complet

livekit.prod.yaml :

```

yaml

# livekit.prod.yaml
port: 443
rtc:
  tcp_port: 443
  udp_port: 443

```

```

port_range_start: 50000
port_range_end: 60000
use_external_ip: true
enable_loopback_candidate: false

keys:
# Générez avec : livekit-server generate-keys
YOUR_API_KEY: YOUR_API_SECRET

redis:
address: redis:6379
password: "votre-mot-de-passe-redis"
db: 0

turn:
enabled: true
domain: "votre-domaine.com"
tls_port: 5349
udp_port: 3478
external_tls: true
cert_file: "/certs/cert.pem"
key_file: "/certs/key.pem"

logging:
level: info
json: true

# Monitoring
prometheus:
enabled: true
port: 9090

```

2.2 Docker Compose pour production

docker-compose.prod.yml :

```

yaml
version: '3.8'

services:
livekit:
image: livekit/livekit:latest
container_name: livekit-prod

```

```

restart: always
ports:
  - "443:443"
  - "443:443/udp"
  - "3478:3478/udp" # TURN UDP
  - "5349:5349"     # TURN TLS
  - "9090:9090"     # Prometheus (optionnel)
volumes:
  - ./livekit.prod.yaml:/livekit.yaml
  - ./certs:/certs:ro
environment:
  - LIVEKIT_CONFIG=/livekit.yaml
  - TZ=Europe/Paris
networks:
  - livekit-network
depends_on:
  redis:
    condition: service_healthy
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:7880/rtc"]
  interval: 30s
  timeout: 10s
  retries: 3

redis:
  image: redis:7-alpine
  container_name: livekit-redis-prod
  restart: always
  command: redis-server --requirepass votre-mot-de-passe-complexe --appendonly yes
  volumes:
    - redis-data:/data
networks:
  - livekit-network
healthcheck:
  test: ["CMD", "redis-cli", "ping"]
  interval: 30s
  timeout: 10s
  retries: 3

# Nginx comme reverse proxy (optionnel)
nginx:
  image: nginx:alpine
  container_name: nginx-proxy

```

```

restart: always
ports:
  - "80:80"
  - "443:443"
volumes:
  - ./nginx.conf:/etc/nginx/nginx.conf:ro
  - ./certs:/etc/nginx/certs:ro
depends_on:
  - livekit
networks:
  - livekit-network

volumes:
  redis-data:

networks:
  livekit-network:
    driver: bridge

```

2.3 Configuration Nginx (optionnel)

nginx.conf :

```

nginx

events {
    worker_connections 1024;
}

http {
    upstream livekit {
        server livekit:7881;
    }

    server {
        listen 80;
        server_name votre-domaine.com;
        return 301 https://$server_name$request_uri;
    }

    server {
        listen 443 ssl http2;
        server_name votre-domaine.com;

```

```

ssl_certificate /etc/nginx/certs/cert.pem;
ssl_certificate_key /etc/nginx/certs/key.pem;

location / {
    proxy_pass http://livekit;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}
}

```

3. Déploiement en Production

3.1 Préparation

bash

```

# 1. Générer les clés API
docker run --rm livekit/livekit:latest livekit-server generate-keys

# 2. Créer la structure de dossiers
mkdir -p livekit-prod/{certs,data,config}
cd livekit-prod

# 3. Placer vos certificats SSL
# cert.pem et key.pem dans le dossier certs/

# 4. Créer le fichier de configuration
nano config/livekit.yaml

# 5. Démarrer
docker-compose -f docker-compose.prod.yml up -d

```

3.2 Script de déploiement automatisé

deploy.sh :

```

bash
#!/bin/bash

# Variables
DOMAIN="votre-domaine.com"
EMAIL="admin@votre-domaine.com"

# Mettre à jour la configuration
sed -i "s/votre-domaine.com/$DOMAIN/g" livekit.prod.yaml
sed -i "s/votre-mot-de-passe-redis/$REDIS_PASSWORD/g" livekit.prod.yaml

# Arrêter les containers existants
docker-compose -f docker-compose.prod.yml down

# Nettoyer les anciens volumes (optionnel)
# docker volume prune -f

# Démarrer
docker-compose -f docker-compose.prod.yml up -d --build

# Vérifier le statut
sleep 10
docker ps
docker logs livekit-prod --tail 50

```

3.3 Variables d'environnement (alternative)

Créez un fichier **.env** :

```

env
# .env
LIVEKIT_API_KEY=APIxxxxxxxxxxxxx
LIVEKIT_API_SECRET=xxxxxxxxxxxxxxxxxx
REDIS_PASSWORD=xxxxxxxxxxxxxx
DOMAIN=votre-domaine.com
TURN_SECRET=xxxxxxxxxxxxxx

```

4. Gestion et Maintenance

4.1 Commandes utiles

```
bash

# Vérifier les logs
docker logs -f livekit-prod

# Scale (si nécessaire)
docker-compose -f docker-compose.prod.yml up -d --scale livekit=3

# Backup Redis
docker exec livekit-redis-prod redis-cli SAVE
docker cp livekit-redis-prod:/data/dump.rdb ./backup/

# Mettre à jour
docker-compose -f docker-compose.prod.yml pull
docker-compose -f docker-compose.prod.yml up -d

# Monitoring
curl http://localhost:9090/metrics # Prometheus
curl http://localhost:7881/rtc      # Health check
```

4.2 Configuration du monitoring

Ajoutez à `livekit.prod.yaml`:

```
yaml

prometheus:
  enabled: true
  port: 9090
```

5. Sécurité

5.1 Meilleures pratiques

1. **Changez les mots de passe par défaut**
2. **Utilisez HTTPS uniquement**
3. **Restreignez l'accès réseau**

4. Mettez à jour régulièrement
5. Monitorer les logs

5.2 Firewall

```
bash

# Ouvrir uniquement les ports nécessaires
sudo ufw allow 443/tcp
sudo ufw allow 443/udp
sudo ufw allow 3478/udp
sudo ufw allow 5349/tcp
```

6. Dépannage

Problèmes courants :

1. WebRTC ne fonctionne pas
 - Vérifiez les certificats SSL
 - Configurez correctement TURN
 - Ouvrez les ports UDP
2. Connexions refusées

```
bash

# Tester la connectivité
curl -v https://votre-domaine.com/rtc
telnet votre-domaine.com 443
```

3. Performances médiocres

- Augmentez les limites système
- Utilisez Redis persistant
- Scalez horizontalement



Ressources supplémentaires

- Documentation officielle LiveKit
- Dépôt GitHub LiveKit
- Guide de déploiement avancé

Ce tutoriel couvre les bases du déploiement self-hosted de LiveKit avec Docker. Adaptez les configurations selon vos besoins spécifiques en production.

