Guida al Deployment di Fluxer: Microservizi, SeaweedFS S3 e Audio WebRTC con LiveKit SFU

Guida al Deployment di Fluxer: Microservizi, SeaweedFS S3 e Audio WebRTC con LiveKit SFU

Nel contesto delle infrastrutture di comunicazione riservate e collaborative, la dipendenza da piattaforme commerciali SaaS esterne (quali Discord, Slack o Microsoft Teams) solleva problemi critici di sovranità sul dato, aderenza normativa (GDPR/NIS2) e rischio di interruzione del servizio da terze parti.

Fluxer costituisce un'architettura self-hosted ad alte prestazioni per la messaggistica istantanea e la comunicazione vocale/video di gruppo. La piattaforma adotta un'ingegnerizzazione a microservizi pensata per installazioni on-premises scalabili e isolate: il database relazionale (**PostgreSQL 16**), il layer di caching volatile in-memory (**Valkey/ Redis**), lo storage a oggetti compatibile con le API S3 (**SeaweedFS**) per la persistenza di avatar, media e allegati, e l'engine multimediale real-time (**LiveKit SFU**).

In questo articolo viene documentata la procedura di installazione da zero, con particolare attenzione all'architettura per l'attraversamento trasparente dei firewall aziendali più restrittivi: **zero porte non-standard esposte**. Tutta la piattaforma opera esclusivamente sulla **porta 443** (sia per il traffico WebRTC primario in **443/UDP**, sia per il web e il fallback WebRTC in **443/TCP** multiplexato tramite **Nginx Stream L4 con SNI Preread**).

Condizioni al Contorno e Parametri di Scenario

Per rendere i listati di codice, i file `.env` e le configurazioni YAML immediatamente comprensibili e testabili, l'articolo adotta una serie di parametri di scenario fittizi conformi agli standard IETF per la

documentazione (RFC 2606 per i domini e RFC 5737 per gli indirizzi IP).

Chi intende implementare l'infrastruttura **deve obbligatoriamente sostituire** questi valori con i parametri reali della propria rete prima di avviare i container:

Parametro di Scenario	Valore d'Esempio nell'Articolo	Descrizione e Istruzioni per la Sostituzione
Dominio Principale Fluxer	`fluxer.example.org`	Sostituire con il dominio pubblico principale della piattaforma (record DNS A/AAAA).
Dominio WebRTC / Fallback SNI	`turn.example.org`	Sostituire con il sottodominio dedicato all'instradamento SNI di LiveKit (stesso IP del server).
Indirizzo IP Pubblico Host	`203.0.113.10`	**CRITICO**: Sostituire con l'effettivo indirizzo IP pubblico statico del vostro server host (ricavabile con `curl -s ifconfig.me`). Viene utilizzato dalla direttiva `node_ip` in `livekit.yaml` per generare i candidati ICE WebRTC. Se lasciato fittizio, i client esterni non riusciranno a connettersi ai canali vocali!
Password PostgreSQL	`SuperSecretPostgresPassword2026!`	Sostituire con una password robusta per il database relazionale (es. generabile con `openssl rand -base64 24`).
Segreti di Sessione (JWT/ Cookie)	`c8d4f9b2...` / `7f12e89d...`	Sostituire con stringhe esadecimali casuali a 256 bit per la cifratura delle sessioni (es. `openssl rand -hex 32`).
Credenziali S3 SeaweedFS	`fluxer_storage_key` / `fluxer_storage_secret...`	Sostituire con chiavi univoche per l'accesso programmatico al bucket S3 locale dei media.
	`4b47a61d0ff70bbc...`	

Parametro di Scenario	Valore d'Esempio nell'Articolo	Descrizione e Istruzioni per la Sostituzione
API Secret LiveKit		Sostituire con una chiave esadecimale a 256 bit condivisa tra il backend di Fluxer e LiveKit.

Architettura e Topologia dei Microservizi

I componenti di backend sono confinati all'interno di una rete bridge Docker isolata. Nessun servizio interno (database, storage o porte applicative secondarie) è pubblicato su Internet. L'esterno vede esclusivamente la **porta 443** (TCP e UDP) e la porta 80 (per il redirect HTTP permanente):

```
flowchart TD
    subgraph Clients ["Client e Terminali Operativi"]
        Desktop["Client Desktop / Web App"]
        Mobile["Client Mobile"]
    end

    subgraph Perimeter ["Frontend Perimetrale Anti-Firewall (Solo Porta 443)"]
        StreamRouter["Nginx Stream Router L4\n(Porta 443/TCP con ssl_preread)"]
        NginxHTTP["Nginx Virtual Host HTTPS\n(Porta interna :8443 TCP)"]
        LiveKit["LiveKit SFU Server (Bridge Mode)\n(Porta 443/UDP Media Primario)\n(Porta interna :7881 TCP Fallback)"]
    end

    subgraph CoreStack ["Stack Applicativo Fluxer (Rete Privata)"]
        CaddyEdge["Caddy Ingress Interno\n(:8080)"]
        Gateway["Fluxer Gateway & API\n(REST e WebSocket)"]
        Auth["Modulo Autenticazione e Sessioni"]
        VoiceCtrl["Fluxer LiveKit Controller"]
        Search["Engine Meilisearch\n(Indicizzazione Full-Text)"]
    end

    subgraph DataPersistence ["Livello di Persistenza e Caching"]
        Postgres["PostgreSQL 16\n(Community, Canali, Ruoli, Permessi)"]
        Valkey["Valkey 7.2\n(Session Cache e Bus Pub/Sub)"]
        Seaweed["SeaweedFS Filer + Volume\n(Object Storage Compatibile S3)"]
    end

```

```
    Desktop -->|Web e WSS: fluxer.example.org:443 TCP| StreamRouter
    Mobile -->|Web e WSS: fluxer.example.org:443 TCP| StreamRouter
    Desktop -.->|WebRTC Primario: :443 UDP| LiveKit
    Mobile -.->|WebRTC Primario: :443 UDP| LiveKit
    Desktop -.->|WebRTC Fallback: turn.example.org:443 TCP| StreamRouter
    Mobile -.->|WebRTC Fallback: turn.example.org:443 TCP| StreamRouter

    StreamRouter -->|SNI: fluxer.example.org| NginxHTTP
    StreamRouter -->|SNI: turn.example.org| LiveKit
    NginxHTTP -->|Proxy Pass :8080 TCP| CaddyEdge

    CaddyEdge --> Gateway
    CaddyEdge --> Auth
    CaddyEdge --> Search

    Gateway --> Valkey
    Gateway --> Postgres
    Gateway --> VoiceCtrl
    VoiceCtrl -->|RPC Token Minting| LiveKit
    Gateway --> Seaweed
```

Il Modello Anti-Firewall: 443/UDP e 443/TCP con SNI Preread

In contesti aziendali, bancari o governativi, i firewall e i sistemi DPI impongono policy ferree: **qualsiasi porta diversa da 80 e 443 è bloccata**. Proporre a un utente di collegarsi a una porta non standard (come la 7881 o i classici range WebRTC 50000:60000 UDP) significa condannare l'installazione all'inaccessibilità.

Per risolvere definitivamente il problema, l'architettura applica una convergenza L4/L7 sulla sola porta 443:

1. **Trasporto Primario su 443/UDP (Simulazione QUIC/HTTP-3)**: Poiché gli stack TCP e UDP sono gestiti dal kernel Linux in spazi di socket disgiunti, il container LiveKit (in modalità bridge) pubblica direttamente `443:443/udp`. I firewall perimetrali trattano questo traffico come normale flusso HTTP/3 (QUIC), consentendo il passaggio dei flussi voce a bassissima latenza.
2. **Multiplexing L4 su 443/TCP via Nginx `ssl_preread`**: Sulla porta `443/TCP` ascolta il modulo `stream` di Nginx con la direttiva `ssl_preread on`. Nginx ispeziona il pacchetto TLS *ClientHello* iniziale

estraendo l'estensione SNI (Server Name Indication) **senza decifrare il traffico**:
    - Se l'SNI corrisponde al dominio web (`fluxer.example.org`), il flusso TCP viene passato all'engine HTTP locale su `127.0.0.1:8443` che gestisce TLS, sito web e WebSockets.
    - Se l'SNI corrisponde all'endpoint WebRTC/TURN (`turn.example.org`), il flusso TCP viene inoltrato a livello 4 direttamente al socket interno di fallback di LiveKit (`127.0.0.1:7881`).
3. **Risultato Operativo**: Per il mondo esterno, **esiste esclusivamente la porta 443**. Se una rete blocca integralmente l'UDP, il client WebRTC commuta in trasparenza sul fallback TCP, presentandosi comunque sulla porta standard **443/TCP**.

---

1. Ottimizzazione del Kernel Host e Buffer di Rete

I flussi real-time ad alta densità richiedono buffer di ricezione e trasmissione dimensionati per evitare il drop dei pacchetti UDP sotto carico.

Creare il file `/etc/sysctl.d/99-fluxer-performance.conf`:

```
cat << 'EOF' | sudo tee /etc/sysctl.d/99-fluxer-performance.conf
# Dimensionamento buffer UDP ad alta capacita per LiveKit SFU
net.core.rmem_max = 5000000
net.core.rmem_default = 1048576
net.core.wmem_max = 5000000
net.core.wmem_default = 1048576

# Backlog TCP per picchi di connessioni simultanee
net.core.somaxconn = 4096
net.ipv4.tcp_max_syn_backlog = 8192

# Abilitazione del forwarding di rete per bridge Docker
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
EOF

sudo sysctl --system
```

Installare Docker Engine e il plugin Docker Compose v2:

```
sudo apt-get update && sudo apt-get install -y ca-certificates curl gnupg lsb-release
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update && sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
sudo systemctl enable --now docker
```

---

2. Struttura delle Directory e Configurazione LiveKit

Predisporre la directory principale `/opt/fluxer` e i percorsi per i volumi di persistenza:

```
sudo mkdir -p /opt/fluxer/data/{postgres,valkey,seaweedfs/volume,seaweedfs/filerldb2,meili}
sudo mkdir -p /opt/fluxer/config
cd /opt/fluxer
```

Configurazione di LiveKit (`/opt/fluxer/config/livekit.yaml`)

Definire i parametri operativi per la porta 443 UDP e il socket TCP locale su 7881 (raggiunto internamente da Nginx):

```
PUBLIC_IP=$(curl -s ifconfig.me)
echo "IP Pubblico rilevato: $PUBLIC_IP"
```

Inserire la configurazione in `/opt/fluxer/config/livekit.yaml`:

```
# /opt/fluxer/config/livekit.yaml
port: 7880
bind_addresses:
  - "0.0.0.0"

rtc:
  udp_port: 443
  tcp_port: 7881
  use_external_ip: false
  # NOTA CRITICA:
  # Inserire l'indirizzo IP pubblico statico reale del server (ricavabile con curl ifconfig.me).
  # NON lasciare 203.0.113.10: LiveKit comunica questo IP ai client WebRTC!
  node_ip: 203.0.113.10

turn:
  enabled: false

keys:
  fluxer: 4b47a61d0ff70bbc9d3dc92d87ac3912be77a42db3dc2c11c37ca2a49833ac9b

logging:
  level: info
```

---

3. Variabili d'Ambiente (`/opt/fluxer/.env`)

Configurare il file `.env` impostando i segreti crittografici e i domini di produzione:

```
cat << 'EOF' > /opt/fluxer/.env
# Identificativi di Dominio ed Endpoint
FLUXER_DOMAIN=fluxer.example.org
FLUXER_PUBLIC_URL=https://fluxer.example.org
FLUXER_API_URL=https://fluxer.example.org/api

# Segreti di Sessione e Token (Generare con: openssl rand -hex 32)
JWT_SECRET=c8d4f9b2a7e1358d601f2e4b89ca0517d3b9e4a816c20573e8fa912567bcde34
COOKIE_SECRET=7f12e89d604b1c5a932e48fa017bd6c39a852e147f0bc63d8912e54a7bc901ef

# Database PostgreSQL 16
POSTGRES_DB=fluxer
POSTGRES_USER=fluxer
POSTGRES_PASSWORD=SuperSecretPostgresPassword2026!
```

```
POSTGRES_HOST=postgres
POSTGRES_PORT=5432

# Caching Valkey
VALKEY_HOST=valkey
VALKEY_PORT=6379

# Storage SeaweedFS S3
SEAWEEDFS_FILER=seaweedfs:8888
SEAWEEDFS_S3_ENDPOINT=http://seaweedfs:8333
S3_ACCESS_KEY=fluxer_storage_key
S3_SECRET_KEY=fluxer_storage_super_secret_key_2026
S3_BUCKET=fluxer-media

# Indicizzazione Meilisearch
MEILI_MASTER_KEY=MeiliSearchSecureMasterKey2026!
MEILI_HOST=http://meilisearch:7700

# Parametri Integrazione LiveKit SFU (Coerenti con livekit.yaml)
FLUXER_LIVEKIT_URL=https://fluxer.example.org/livekit
FLUXER_LIVEKIT_INTERNAL_URL=http://livekit:7880
FLUXER_LIVEKIT_API_KEY=fluxer
FLUXER_LIVEKIT_API_SECRET=4b47a61d0ff70bbc9d3dc92d87ac3912be77a42db3dc2c11c37ca2a49833ac9b
FLUXER_LIVEKIT_UDP_PORT=443
EOF
```

---

4. Orchestrazione dello Stack (`docker-compose.yaml`)

Tutti i microservizi operano sulla rete bridge isolata `fluxer-net`. Notare come LiveKit pubblichi verso l'esterno esclusivamente la porta `443/udp`, mentre le porte TCP `7880` (segnalazione) e `7881` (fallback ICE) sono vincolate unicamente sull'interfaccia locale `127.0.0.1` per Nginx:

```
# /opt/fluxer/docker-compose.yaml
services:
  # Ingress Interno Caddy
  edge:
    image: caddy:2-alpine
    container_name: fluxer-edge
    restart: unless-stopped
    ports:
      - "127.0.0.1:8080:8080"
    volumes:
      - ./config/Caddyfile:/etc/caddy/Caddyfile:ro
```

```
    networks:
      - fluxer-net
    depends_on:
      - api

  # Core Service: API, Gateway e Logica di Business
  api:
    image: ghcr.io/fluxerapp/server:latest
    container_name: fluxer-api
    restart: unless-stopped
    env_file: .env
    networks:
      - fluxer-net
    depends_on:
      postgres:
        condition: service_healthy
      valkey:
        condition: service_started
      seaweedfs:
        condition: service_started
      livekit:
        condition: service_started

  # Database Relazionale: PostgreSQL 16
  postgres:
    image: postgres:16-alpine
    container_name: fluxer-postgres
    restart: unless-stopped
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      - ./data/postgres:/var/lib/postgresql/data
    networks:
      - fluxer-net
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER} -d ${POSTGRES_DB}"]
      interval: 5s
      timeout: 5s
      retries: 5

  # Cache In-Memory: Valkey 7.2
  valkey:
    image: valkey/valkey:7.2-alpine
    container_name: fluxer-valkey
    restart: unless-stopped
```

```
    command: valkey-server --save 60 1 --loglevel warning
    volumes:
      - ./data/valkey:/data
    networks:
      - fluxer-net

  # Media Storage: SeaweedFS (Filer e Gateway S3)
  seaweedfs:
    image: chrislusf/seaweedfs:latest
    container_name: fluxer-seaweedfs
    restart: unless-stopped
    command: "server -dir=/data -s3 -filer -volume.max=0"
    volumes:
      - ./data/seaweedfs/volume:/data
      - ./data/seaweedfs/filerldb2:/data/filerldb2
    networks:
      - fluxer-net

  # Engine di Ricerca Full-Text: Meilisearch
  meilisearch:
    image: getmeili/meilisearch:v1.6
    container_name: fluxer-meilisearch
    restart: unless-stopped
    environment:
      MEILI_MASTER_KEY: ${MEILI_MASTER_KEY}
      MEILI_NO_ANALYTICS: "true"
    volumes:
      - ./data/meili:/meili_data
    networks:
      - fluxer-net

  # Media Server WebRTC SFU: LiveKit (Bridge Mode Anti-Firewall)
  livekit:
    image: livekit/livekit-server:latest
    container_name: fluxer-livekit
    restart: unless-stopped
    command: ["--config", "/etc/livekit.yaml"]
    volumes:
      - ./config/livekit.yaml:/etc/livekit.yaml:ro
    ports:
      - "127.0.0.1:7880:7880"   # Segnalazione interna verso Nginx
      - "443:443/udp"           # Media WebRTC Primario (Esterno)
      - "127.0.0.1:7881:7881"   # Media WebRTC TCP Fallback verso Nginx Stream
    networks:
      - fluxer-net

networks:
```

```
  fluxer-net:
    name: fluxer-net
    driver: bridge
```

---

5. Configurazione Nginx: Stream L4 SNI Preread e Virtual Host

La configurazione del reverse proxy Nginx si articola su due livelli: il router L4 nel blocco `stream` e i virtual host HTTPS nel blocco `http`.

5.1 Il Blocco `stream` in `/etc/nginx/nginx.conf`

Nel file principale `/etc/nginx/nginx.conf`, aggiungere il blocco `stream` (allo stesso livello di `http`):

```
# Inserire in /etc/nginx/nginx.conf (a livello root, fuori da http { })
stream {
    # Mappa SNI: instrada i pacchetti TLS in base all'hostname richiesto dal client
    map $ssl_preread_server_name $backend_pool {
        turn.example.org    127.0.0.1:7881; # LiveKit WebRTC TCP Fallback
        default             127.0.0.1:8443; # Virtual Host HTTPS locale di Nginx
    }

    server {
        listen 443;
        listen [::]:443;
        proxy_pass $backend_pool;
        ssl_preread on;
        proxy_connect_timeout 10s;
        proxy_timeout 24h;
    }
}
```

5.2 Il Virtual Host HTTPS in `/etc/nginx/sites-available/fluxer.example.org.conf`

Poiché la porta `443/TCP` esterna è gestita dal modulo `stream`, il virtual host HTTP/HTTPS risiede in ascolto locale sulla porta `127.0.0.1:8443`:

```
# /etc/nginx/sites-available/fluxer.example.org.conf
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

# Redirect HTTP su porta 80
server {
    listen 80;
    server_name fluxer.example.org turn.example.org;
    return 301 https://$host$request_uri;
}

# Virtual Host HTTPS Interno (Riceve da Nginx Stream su 127.0.0.1:8443)
server {
    listen 127.0.0.1:8443 ssl http2;
    server_name fluxer.example.org;

    ssl_certificate /etc/letsencrypt/live/fluxer.example.org/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/fluxer.example.org/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    client_max_body_size 100M;

    # Canale di Segnalazione LiveKit SFU (WebSockets)
    location ^~ /livekit/ {
        proxy_pass http://127.0.0.1:7880/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 86400s;
        proxy_send_timeout 86400s;
    }

    # Ingress Principale Piattaforma Fluxer
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $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;
```

```
        proxy_buffering off;
        proxy_read_timeout 3600s;
    }
}
```

Abilitare la configurazione e riavviare Nginx:

```
sudo ln -sf /etc/nginx/sites-available/fluxer.example.org.conf /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl restart nginx
```

---

6. Procedura di Collaudo e Validazione Funzionale

Avviare la suite:

```
cd /opt/fluxer && sudo docker compose up -d
```

Eseguire la sequenza di verifiche strumentali (sostituendo `fluxer.example.org` e `turn.example.org` con i vostri domini reali):

```
# 1. Verifica dell'Ingress Web HTTPS sulla porta standard 443 TCP
curl -I https://fluxer.example.org/_health
# Risultato atteso: HTTP/2 200 OK

# 2. Verifica risposta dell'endpoint di segnalazione LiveKit
curl -I https://fluxer.example.org/livekit/
# Risultato atteso: HTTP/1.1 200 OK (o WebSocket 400 Bad Request se privo di header di upgrade)

# 3. Verifica ascolto porta 443/UDP (WebRTC Primario)
nc -zvu fluxer.example.org 443
# Risultato atteso: Connection to fluxer.example.org 443 port [udp/https] succeeded!

# 4. Verifica instradamento L4 SNI Preread sulla porta standard 443/TCP
openssl s_client -connect turn.example.org:443 -servername turn.example.org
# Risultato atteso: Connessione stabilita con il backend LiveKit TCP senza errori di reset
```

---

7. Strategia di Backup e Manutenzione

Pianificare un processo notturno per l'archiviazione consistente dei database e dei dati binari:

```
#!/usr/bin/env bash
set -euo pipefail
BACKUP_DIR="/backup/fluxer/$(date +%F)"
mkdir -p "$BACKUP_DIR"

# Dump logico consistente di PostgreSQL
docker exec -t fluxer-postgres pg_dumpall -U fluxer > "$BACKUP_DIR/
fluxer_db.sql"

# Snapshot metadati e volumi SeaweedFS
rsync -aAX /opt/fluxer/data/seaweedfs/filerldb2/ "$BACKUP_DIR/seaweedfs_meta/"
rsync -aAX /opt/fluxer/data/seaweedfs/volume/ "$BACKUP_DIR/seaweedfs_data/"

# Compressione archivio cifrato
tar -czvf "$BACKUP_DIR.tar.gz" -C "$BACKUP_DIR" .
rm -rf "$BACKUP_DIR"
```

Grazie alla combinazione della modalità bridged, del trasporto primario su 443/UDP e del multiplexing L4 con SNI Preread sulla sola porta standard 443/TCP, l'infrastruttura Fluxer garantisce comunicazioni vocali e collaborative resilienti anche dietro i firewall perimetrali più aggressivi, eliminando del tutto la necessità di aprire porte esotiche o non-standard.

Fluxer Icon

Canale di discussione e supporto

Canale di discussione e supporto. Per confrontarti su configurazioni o altro entra su fluxer.dreadful.work (https://fluxer.dreadful.work/invite/dreadful).