TerminalX Containerized Nginx Deployment Guide

This guide sets up TerminalX with a **containerized nginx reverse proxy** where all configuration is managed through environment variables in docker-compose.yml and SSL certificates are stored on the **host filesystem**.

🎯 Architecture Overview

```
Internet → Host Firewall → Docker Host (nginx container) → Internal Network
                    ├── Port 80/443 (HTTPS FQDN)
                    ├── Port 8087 (Direct TerminalX)
                    └── Port 3000 (Direct SFTP)
                         ↓
                    terminalx-network (bridge)
                    ├── nginx container (reverse proxy)
                    ├── terminalx container
                    └── sftp-browser container
```

📋 Final Access Methods

After deployment, users can access services via:

- 🔒 **HTTPS FQDN**: `https://terminalx.yourdomain.com` → TerminalX (SSL secured)
- 🖥️ **Direct TerminalX**: `http://YOUR_SERVER_IP:8087` → TerminalX
- 📁 **Direct SFTP**: `http://YOUR_SERVER_IP:3000` → SFTP Server

🔧 Key Features

Container Architecture

- ✅ Nginx runs as a container (not host service)
- ✅ All configuration via environment variables
- ✅ SSL certificates stored on host filesystem
- ✅ Automatic certificate renewal
- ✅ No external nginx installation required

Environment Variable Configuration

- ✅ Domain, SSL paths, backend services configurable
- ✅ Performance tuning via environment variables

- ✅ Rate limiting and security settings configurable
- ✅ No manual config file editing required

Host Filesystem Certificate Storage

- ✅ Certificates stored in `/etc/ssl/` (host filesystem)
- ✅ Persistent across container restarts
- ✅ Easy backup and management
- ✅ Standard Linux certificate locations

🚀 Quick Start Deployment

Step 1: Update Configuration

Edit the SSL setup script:

```bash
# Download and edit the setup script
curl -O https://your-script-location/ssl-setup.sh
chmod +x ssl-setup.sh

# Edit these variables:
vim ssl-setup.sh
```

Update these lines in `ssl-setup.sh`:

```bash
DOMAIN="terminalx.yourdomain.com"    # Your actual domain
EMAIL="admin@yourdomain.com"         # Your actual email
```

Step 2: DNS Configuration

Create DNS A record:

```dns
terminalx.yourdomain.com.    IN    A    YOUR_SERVER_IP
```

Validate DNS:

```bash
```

```
sudo ./ssl-setup.sh --validate-dns
```

Step 3: Choose Certificate Type

Option A: Let's Encrypt (Production)

```bash
sudo ./ssl-setup.sh --letsencrypt
```

Option B: Let's Encrypt (Staging - for testing)

```bash
sudo ./ssl-setup.sh --letsencrypt-staging
```

Option C: Self-Signed (Development)

```bash
sudo ./ssl-setup.sh --self-signed
```

Step 4: Verify Deployment

```bash
# Check services
docker-compose ps

# Test HTTPS access
curl -I https://terminalx.yourdomain.com

# Test direct access
curl -I http://YOUR_SERVER_IP:8087
curl -I http://YOUR_SERVER_IP:3000
```

📁 Directory Structure

```
terminalx-deployment/
├── docker-compose.yml          # Container orchestration with env vars
├── ssl-setup.sh                # SSL certificate management
├── renew-certificates.sh       # Auto-renewal script
```

```
│
├── nginx/                    # Nginx configuration templates
│   ├── nginx.conf.template       # Main nginx config template
│   └── conf.d/
│       └── terminalx.conf.template # Site-specific template
│
├── logs/nginx/               # Nginx logs (host mounted)
├── letsencrypt/              # Let's Encrypt webroot & logs
│   ├── webroot/              # ACME challenge files
│   └── logs/                 # Certbot logs
│
└── /etc/ssl/                 # Host filesystem certificates
    ├── certs/
    │   ├── terminalx.crt         # SSL certificate
    │   └── terminalx-chain.crt   # Certificate chain
    └── private/
        └── terminalx.key         # Private key
```

⚙️ Environment Variable Configuration

All configuration is managed through environment variables in `docker-compose.yml`:

Domain & SSL Configuration

```yaml
environment:
  - DOMAIN=terminalx.yourdomain.com
  - SERVER_NAME=terminalx.yourdomain.com
  - SSL_CERT_PATH=/etc/ssl/certs/terminalx.crt
  - SSL_KEY_PATH=/etc/ssl/private/terminalx.key
  - SSL_CHAIN_PATH=/etc/ssl/certs/terminalx-chain.crt
```

Backend Services

```yaml
environment:
  - TERMINALX_BACKEND=terminalx:8087
  - SFTP_BACKEND=sftp-browser:3000
```

Performance Tuning

```yaml
environment:
  - WORKER_PROCESSES=auto
  - WORKER_CONNECTIONS=1024
  - CLIENT_MAX_BODY_SIZE=100M
```

Security Settings

```yaml
environment:
  - LOGIN_RATE_LIMIT=5r/m
  - API_RATE_LIMIT=10r/s
```

🔐 SSL Certificate Management

Certificate Locations (Host Filesystem)

- **Certificate**: `/etc/ssl/certs/terminalx.crt` → Symlink to Let's Encrypt
- **Private Key**: `/etc/ssl/private/terminalx.key` → Symlink to Let's Encrypt
- **Chain**: `/etc/ssl/certs/terminalx-chain.crt` → Symlink to Let's Encrypt
- **Let's Encrypt**: `/etc/letsencrypt/live/yourdomain.com/`

Auto-Renewal Process

1. **Cron job** runs twice daily (3 AM & 3 PM)
2. **Certbot** checks for renewal needed
3. **Symlinks** updated to new certificates
4. **Nginx** reloaded with new certificates

Manual Operations

```bash
```

```bash
# Test certificate renewal
sudo ./ssl-setup.sh --renew

# Check certificate expiry
sudo openssl x509 -in /etc/ssl/certs/terminalx.crt -noout -dates

# Test SSL configuration
sudo ./ssl-setup.sh --test

# View renewal logs
tail -f letsencrypt/logs/renewal.log
```

🛠 Container Management

Service Control

```bash
# View all services
docker-compose ps

# View service logs
docker-compose logs nginx
docker-compose logs terminalx
docker-compose logs sftp-browser

# Restart specific service
docker-compose restart nginx

# Update configuration and restart
docker-compose down
docker-compose up -d
```

Nginx Container Operations

```bash
```

```bash
# Test nginx configuration
docker exec terminalx-nginx nginx -t

# Reload nginx (without restart)
docker exec terminalx-nginx nginx -s reload

# View nginx processes
docker exec terminalx-nginx ps aux

# Access nginx container
docker exec -it terminalx-nginx /bin/sh
```

Environment Variable Updates

```bash
# Edit docker-compose.yml
vim docker-compose.yml

# Apply changes
docker-compose down
docker-compose up -d

# Verify new configuration
docker exec terminalx-nginx env | grep -E "(DOMAIN|SSL_|BACKEND)"
```

🔍 Troubleshooting

Container Issues

```bash
```

```bash
# Check container status
docker-compose ps

# View container logs
docker-compose logs --tail=50 nginx

# Check container resource usage
docker stats

# Inspect container configuration
docker inspect terminalx-nginx
```

SSL Certificate Issues

```bash
# Verify certificate files exist
sudo ls -la /etc/ssl/certs/terminalx.crt
sudo ls -la /etc/ssl/private/terminalx.key

# Check certificate validity
sudo openssl x509 -in /etc/ssl/certs/terminalx.crt -text -noout

# Test SSL connection
openssl s_client -connect yourdomain.com:443 -servername yourdomain.com

# Force certificate renewal
sudo ./ssl-setup.sh --letsencrypt
```

Network Connectivity Issues

```bash
```

```bash
# Test internal container connectivity
docker exec terminalx-nginx ping terminalx
docker exec terminalx-nginx ping sftp-browser

# Check port bindings
docker port terminalx-nginx

# Test external access
curl -I http://YOUR_SERVER_IP:8087
curl -I http://YOUR_SERVER_IP:3000

# Check firewall rules
sudo ufw status
sudo iptables -L
```

Configuration Issues

```bash
# Validate nginx configuration
docker exec terminalx-nginx nginx -t

# Check environment variable substitution
docker exec terminalx-nginx cat /etc/nginx/nginx.conf | head -20
docker exec terminalx-nginx cat /etc/nginx/conf.d/terminalx.conf | head -20

# Verify template processing
docker exec terminalx-nginx env | grep -E "(DOMAIN|SSL_|BACKEND)"
```

📊 Monitoring & Maintenance

Log Monitoring

```bash
```

```
# Real-time nginx logs
tail -f logs/nginx/access.log
tail -f logs/nginx/error.log

# WebSocket connection logs
tail -f logs/nginx/websocket.log

# Certificate renewal logs
tail -f letsencrypt/logs/renewal.log

# All container logs
docker-compose logs -f
```

Health Checks

```bash
# Built-in health endpoints
curl http://YOUR_SERVER_IP:8087/health    # TerminalX health
curl http://YOUR_SERVER_IP:3000/health    # SFTP health

# SSL certificate check
curl -I https://yourdomain.com/health

# Service availability check
./ssl-setup.sh --test
```

Performance Monitoring

```bash
# Container resource usage
docker stats terminalx-nginx terminalx sftp-server

# Nginx connection statistics
docker exec terminalx-nginx nginx -s reload  # Reset counters
docker exec terminalx-nginx cat /proc/net/sockstat
```

🔄 Backup & Recovery

Backup Important Files

```bash
```

```bash
#!/bin/bash
# Backup script
backup_date=$(date +%Y%m%d_%H%M%S)
backup_dir="backup_$backup_date"

mkdir -p "$backup_dir"

# Backup configuration
cp docker-compose.yml "$backup_dir/"
cp -r nginx/ "$backup_dir/"
cp ssl-setup.sh "$backup_dir/"
cp renew-certificates.sh "$backup_dir/"

# Backup certificates (as root)
sudo cp -r /etc/ssl/certs/terminalx* "$backup_dir/"
sudo cp -r /etc/ssl/private/terminalx* "$backup_dir/"
sudo cp -r /etc/letsencrypt/ "$backup_dir/"

# Create archive
tar -czf "terminalx-backup-$backup_date.tar.gz" "$backup_dir"
rm -rf "$backup_dir"

echo "Backup created: terminalx-backup-$backup_date.tar.gz"
```

Recovery Process

```bash
```

```
# Extract backup
tar -xzf terminalx-backup-YYYYMMDD_HHMMSS.tar.gz

# Restore certificates (as root)
sudo cp backup_*/terminalx* /etc/ssl/certs/
sudo cp backup_*/terminalx* /etc/ssl/private/
sudo cp -r backup_*/letsencrypt/* /etc/letsencrypt/

# Restore configuration
cp backup_*/docker-compose.yml .
cp -r backup_*/nginx/ .

# Restart services
docker-compose down
docker-compose up -d
```

🚀 Advanced Configuration

Custom Environment Variables

Add to docker-compose.yml nginx service:

```yaml
environment:
  # Custom rate limits
  - UPLOAD_RATE_LIMIT=2r/s
  - DOWNLOAD_RATE_LIMIT=10r/s

  # Custom timeouts
  - PROXY_TIMEOUT=120s
  - WEBSOCKET_TIMEOUT=7200s

  # Custom security headers
  - CUSTOM_HEADER_NAME=Custom-Value
```

Multiple Domain Support

```yaml
environment:
  - DOMAIN=terminalx.domain1.com,terminalx.domain2.com
  - SERVER_NAME=terminalx.domain1.com terminalx.domain2.com
```

Performance Tuning

```yaml
environment:
  - WORKER_PROCESSES=4          # Match CPU cores
  - WORKER_CONNECTIONS=2048     # Increase for high traffic
  - CLIENT_MAX_BODY_SIZE=500M   # Increase for large uploads
  - PROXY_BUFFERING=off         # Disable for real-time apps
```

📞 Support & Troubleshooting

Quick Diagnostics

```bash
# Run comprehensive health check
sudo ./ssl-setup.sh --test && \
docker-compose ps && \
curl -I https://yourdomain.com && \
curl -I http://YOUR_SERVER_IP:8087 && \
echo "All systems operational"
```

Common Issues & Solutions

1. **Containers not starting**: Check `docker-compose logs`
2. **Certificate errors**: Run `sudo ./ssl-setup.sh --letsencrypt`
3. **Network issues**: Verify firewall and DNS configuration
4. **Permission errors**: Ensure script runs with `sudo`
5. **WebSocket failures**: Check nginx WebSocket configuration

Getting Help

1. **Check logs first**: `docker-compose logs nginx`
2. **Verify configuration**: `docker exec terminalx-nginx nginx -t`
3. **Test connectivity**: Use curl commands above
4. **Review environment**: `docker exec terminalx-nginx env`

🎉 Benefits of This Architecture

✅ **Container-native**: Everything runs in containers ✅ **Environment-driven**: No manual config editing ✅ **Host certificate storage**: Persistent and manageable ✅ **Auto-renewal**: Set-and-forget certificate management ✅ **Production-ready**: Security, performance, and monitoring included ✅ **Easy deployment**: Single script setup ✅ **Easy maintenance**: Simple container management

This setup provides enterprise-grade reverse proxy functionality while maintaining the simplicity and reliability of containerized deployments.