Complete Deployment Guide

This comprehensive guide walks you through deploying the gaming-themed investor website on your Proxmox server using LXC containers and Docker.

Overview

The deployment architecture consists of:
- **Proxmox LXC Container**: Lightweight virtualization for the application stack
- **Docker Compose**: Orchestrates multiple services (Next.js, PostgreSQL, Nginx, Redis)
- **Nginx Reverse Proxy**: Handles SSL termination and load balancing
- **PostgreSQL Database**: Stores user data, sessions, and investment information
- **Redis Cache**: Session storage and caching (optional but recommended)

Prerequisites

Before starting the deployment:

- [ ] Proxmox VE 7.x or higher installed and configured
- [ ] At least 8GB RAM and 50GB storage available
- [ ] Network access to download container templates and Docker images
- [ ] Basic familiarity with Linux command line
- [ ] Your gaming-themed investor website code ready at `/home/ubuntu/arcade_investor_site`

Phase 1: Proxmox LXC Container Setup

Step 1: Create and Configure LXC Container

Follow the detailed instructions in `PROXMOX_LXC_SETUP.md` to:

1. Download Ubuntu 22.04 template
2. Create LXC container with Docker support
3. Configure container for Docker (nesting, keyctl, etc.)
4. Install Docker and Docker Compose
5. Configure networking and firewall

Quick Setup Commands:

```
# On Proxmox host - Create container
pct create 105 local:vztmpl/ubuntu-22.04-standard_22.04-1_amd64.tar.zst \
  --hostname investor-website \
  --storage local-lvm \
  --rootfs 20G \
  --memory 4096 \
  --swap 512 \
  --cores 2 \
  --net0 name=eth0,bridge=vmbr0,ip=dhcp \
  --features nesting=1,keyctl=1 \
  --unprivileged 1

# Configure for Docker
echo "lxc.apparmor.profile: unconfined" >> /etc/pve/lxc/105.conf
echo "lxc.cgroup.devices.allow: a" >> /etc/pve/lxc/105.conf
echo "lxc.cap.drop:" >> /etc/pve/lxc/105.conf

# Start container
pct start 105
```

Step 2: Prepare Container Environment

```
# Access container
pct exec 105 -- bash

# Update system and install Docker (see PROXMOX_LXC_SETUP.md for full commands)
apt-get update && apt-get upgrade -y
# ... Docker installation commands ...

# Create deployment user
adduser deploy
usermod -aG docker deploy
usermod -aG sudo deploy
```

Phase 2: Transfer Deployment Files

Step 3: Copy Deployment Configuration

From your Proxmox host or development machine:

```
# Copy deployment files to container
scp -r ~/proxmox_deployment deploy@<container-ip>:/home/deploy/
scp -r ~/arcade_investor_site deploy@<container-ip>:/home/deploy/

# Or if accessing from within Proxmox host
pct push 105 ~/proxmox_deployment /home/deploy/proxmox_deployment -user deploy -group d
eploy
pct push 105 ~/arcade_investor_site /home/deploy/arcade_investor_site -user deploy -
group deploy
```

Step 4: Set Up Environment Configuration

```
# Access container as deploy user
pct exec 105 -- su - deploy

# Navigate to deployment directory
cd /home/deploy/proxmox_deployment

# Copy environment template
cp docker/.env.example docker/.env

# Edit environment variables
nano docker/.env
```

Critical Environment Variables to Configure:

```
# Database Configuration
POSTGRES_DB=investor_db
POSTGRES_USER=investor_user
POSTGRES_PASSWORD=your_secure_database_password

# NextAuth Configuration
NEXTAUTH_SECRET=your_secure_nextauth_secret_key
NEXTAUTH_URL=https://investor.local

# Domain Configuration
DOMAIN=investor.local

# Email Configuration (for magic links)
EMAIL_SERVER_HOST=smtp.gmail.com
EMAIL_SERVER_PORT=587
EMAIL_SERVER_USER=your_email@gmail.com
EMAIL_SERVER_PASSWORD=your_gmail_app_password
EMAIL_FROM=noreply@investor.local

# OAuth Configuration (optional)
GOOGLE_CLIENT_ID=your_google_client_id
GOOGLE_CLIENT_SECRET=your_google_client_secret
```

Phase 3: SSL Certificate Setup

Step 5: Configure SSL Certificates

Choose one of the following methods:

Option A: Using mkcert (Recommended for Internal Use)

```
# Install mkcert in container
curl -JLO "https://dl.filippo.io/mkcert/latest?for=linux/amd64"
chmod +x mkcert-v*-linux-amd64
sudo mv mkcert-v*-linux-amd64 /usr/local/bin/mkcert

# Install local CA
mkcert -install

# Generate certificates
cd /home/deploy/proxmox_deployment/nginx
mkdir -p ssl
cd ssl
mkcert investor.local "*.investor.local" localhost 127.0.0.1 ::1

# Rename files to match nginx configuration
mv investor.local+4.pem investor.local.pem
mv investor.local+4-key.pem investor.local-key.pem
```

Option B: Self-Signed Certificates (Automatic)

The nginx container will automatically generate self-signed certificates if none are found.

Option C: Let’s Encrypt (For Public Access)

```
# Install certbot
sudo apt install certbot

# Generate certificates (requires public domain)
sudo certbot certonly --standalone -d your-public-domain.com

# Copy certificates to deployment directory
sudo cp /etc/letsencrypt/live/your-domain/fullchain.pem nginx/ssl/investor.local.pem
sudo cp /etc/letsencrypt/live/your-domain/privkey.pem nginx/ssl/investor.local-key.pem
sudo chown deploy:deploy nginx/ssl/*
```

Phase 4: Application Deployment

Step 6: Prepare Next.js Application

```
# Navigate to your Next.js project
cd /home/deploy/arcade_investor_site

# Ensure next.config.js has standalone output
cat > next.config.js << 'EOF'
/** @type {import('next').NextConfig} */
const nextConfig = {
  output: 'standalone',
  experimental: {
    outputFileTracingRoot: undefined,
  },
  // Add your existing configuration here
}

module.exports = nextConfig
EOF

# Add health check endpoint
cat > healthcheck.js << 'EOF'
const http = require('http');

const options = {
  hostname: 'localhost',
  port: 3000,
  path: '/api/health',
  method: 'GET',
  timeout: 2000
};

const req = http.request(options, (res) => {
  if (res.statusCode === 200) {
    process.exit(0);
  } else {
    process.exit(1);
  }
});

req.on('error', () => process.exit(1));
req.on('timeout', () => {
  req.destroy();
  process.exit(1);
});

req.end();
EOF
```

Step 7: Deploy with Docker Compose

```
# Navigate to deployment directory
cd /home/deploy/proxmox_deployment

# Make scripts executable
chmod +x scripts/*.sh

# Run deployment script
./scripts/deploy.sh
```

Manual Deployment Steps (if script fails):

```
cd docker

# Pull base images
docker-compose pull

# Build containers
docker-compose build --no-cache

# Start services
docker-compose up -d

# Check service status
docker-compose ps

# View logs
docker-compose logs -f
```

Phase 5: Post-Deployment Configuration

Step 8: Verify Deployment

```
# Check all services are running
docker-compose ps

# Test database connection
docker-compose exec postgres psql -U investor_user -d investor_db -c "SELECT ver-
sion();"

# Test Next.js application
curl -k https://localhost/api/health

# Check nginx configuration
docker-compose exec nginx nginx -t
```

Step 9: Configure Host Access

On Proxmox Host:

```
# Add domain to hosts file
echo "$(pct exec 105 -- hostname -I | awk '{print $1}') investor.local" >> /etc/hosts
```

On Your Development Machine:

```
# Add domain to hosts file
echo "<container-ip> investor.local" | sudo tee -a /etc/hosts
```

Step 10: Set Up Monitoring and Logging

```
# Create log monitoring script
cat > /home/deploy/monitor_logs.sh << 'EOF'
#!/bin/bash
cd /home/deploy/proxmox_deployment/docker
docker-compose logs --tail=100 -f
EOF

chmod +x /home/deploy/monitor_logs.sh

# Set up log rotation
sudo tee /etc/logrotate.d/docker-investor << 'EOF'
/var/lib/docker/containers/*/*-json.log {
    rotate 7
    daily
    compress
    size=1M
    missingok
    delaycompress
    copytruncate
}
EOF
```

Phase 6: Security and Backup Setup

Step 11: Configure Firewall

```
# Install and configure UFW
sudo apt install ufw
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw --force enable
```

Step 12: Set Up Automated Backups

```
# Make backup script executable
chmod +x /home/deploy/proxmox_deployment/scripts/backup.sh

# Test backup
./scripts/backup.sh

# Set up cron job for daily backups
crontab -e

# Add this line for daily backup at 2 AM
0 2 * * * /home/deploy/proxmox_deployment/scripts/backup.sh
```

Step 13: Configure SSL Certificate Renewal

```
# Make certificate renewal script executable
chmod +x /home/deploy/proxmox_deployment/scripts/renew_certs.sh

# Test certificate renewal
./scripts/renew_certs.sh

# Set up monthly certificate renewal
crontab -e

# Add this line for monthly renewal
0 3 1 * * /home/deploy/proxmox_deployment/scripts/renew_certs.sh
```

Phase 7: Authentication Setup

Step 14: Configure Authentication

Follow the detailed instructions in `AUTH_SETUP.md` to:

1. Set up OAuth providers (Google)
2. Configure email settings for magic links
3. Test authentication flows
4. Set up investor profiles and KYC verification

Quick Authentication Test:

```
# Access the application
curl -k https://investor.local

# Check authentication endpoints
curl -k https://investor.local/api/auth/providers

# Test database user creation
docker-compose exec postgres psql -U investor_user -d investor_db -c "SELECT * FROM
users;"
```

Phase 8: Performance Optimization

Step 15: Optimize Container Performance

```
# On Proxmox host - optimize container settings
pct set 105 --memory 6144 --swap 1024 --cores 4

# Enable KSM for memory optimization
echo 1 > /sys/kernel/mm/ksm/run

# Configure container I/O optimization
pct set 105 --rootfs local-lvm:vm-105-disk-0,size=30G,cache=writeback
```

Step 16: Database Optimization

```
# Access PostgreSQL container
docker-compose exec postgres psql -U investor_user -d investor_db

# Optimize PostgreSQL settings
ALTER SYSTEM SET shared_buffers = '256MB';
ALTER SYSTEM SET effective_cache_size = '1GB';
ALTER SYSTEM SET maintenance_work_mem = '64MB';
ALTER SYSTEM SET checkpoint_completion_target = 0.9;
ALTER SYSTEM SET wal_buffers = '16MB';
ALTER SYSTEM SET default_statistics_target = 100;

# Restart PostgreSQL to apply changes
\q
docker-compose restart postgres
```

Phase 9: Testing and Validation

Step 17: Comprehensive Testing

```
# Test all endpoints
curl -k https://investor.local/
curl -k https://investor.local/api/health
curl -k https://investor.local/api/auth/providers

# Test database connectivity
docker-compose exec postgres pg_isready -U investor_user -d investor_db

# Test SSL certificate
openssl s_client -connect investor.local:443 -servername investor.local

# Load test (optional)
# Install apache2-utils for ab command
sudo apt install apache2-utils
ab -n 100 -c 10 https://investor.local/
```

Step 18: User Acceptance Testing

1. **Authentication Testing:**
   - [ ] Sign up with email/password
   - [ ] Sign in with Google OAuth
   - [ ] Request magic link via email
   - [ ] Password reset functionality

2. **Investor Features Testing:**
   - [ ] Create investor profile
   - [ ] View investment opportunities
   - [ ] KYC verification process
   - [ ] Investment tracking

3. **Security Testing:**
   - [ ] Rate limiting on auth endpoints
   - [ ] HTTPS redirect working

- [ ] Session management
- [ ] CSRF protection

Phase 10: Production Readiness

Step 19: Production Checklist

- [ ] **Security:**
- [ ] All default passwords changed
- [ ] Firewall configured
- [ ] SSL certificates valid
- [ ] Rate limiting enabled
- [ ] Security headers configured
- [ ] **Performance:**
- [ ] Database optimized
- [ ] Caching configured
- [ ] Static assets optimized
- [ ] Container resources allocated
- [ ] **Monitoring:**
- [ ] Log aggregation set up
- [ ] Health checks configured
- [ ] Backup procedures tested
- [ ] Monitoring alerts configured
- [ ] **Documentation:**
- [ ] Deployment procedures documented
- [ ] Recovery procedures tested
- [ ] User guides created
- [ ] API documentation updated

Step 20: Go-Live Procedures

```
# Final deployment verification
cd /home/deploy/proxmox_deployment/docker
docker-compose ps
docker-compose logs --tail=50

# Create deployment snapshot
pct snapshot 105 "pre-production-$(date +%Y%m%d)"

# Final backup
/home/deploy/proxmox_deployment/scripts/backup.sh

# Monitor initial traffic
tail -f /var/log/nginx/access.log
```

Troubleshooting Guide

Common Issues and Solutions

1. **Container Won’t Start:**
   ```bash
   # Check container configuration
   pct config 105

# Check Proxmox logs
journalctl -u pve-container@105

# Verify nesting is enabled
grep nesting /etc/pve/lxc/105.conf
```

1. **Docker Service Fails:**
   ```bash
   # Check Docker status
   systemctl status docker

# Check Docker logs
journalctl -u docker

# Restart Docker
systemctl restart docker
```

1. **Database Connection Issues:**
   ```bash
   # Check PostgreSQL logs
   docker-compose logs postgres

# Test connection
docker-compose exec postgres psql -U investor_user -d investor_db

# Check network connectivity
docker-compose exec nextjs_app ping postgres
```

1. **SSL Certificate Problems:**
   ```bash
   # Check certificate validity
   openssl x509 -in nginx/ssl/investor.local.pem -text -noout

# Test SSL connection
openssl s_client -connect investor.local:443

# Regenerate certificates
./scripts/renew_certs.sh
```

1. **Performance Issues:**
   ```bash
   # Check container resources
   pct status 105
   ```

```
# Monitor resource usage
docker stats

# Check database performance
docker-compose exec postgres psql -U investor_user -d investor_db -c “SELECT * FROM
pg_stat_activity;”
```

Emergency Recovery Procedures

1. **Service Recovery:**
   ```bash
   # Restart all services
   docker-compose restart

# Rebuild and restart
docker-compose down
docker-compose up –build -d
```

1. **Database Recovery:**
   ```bash
      # Restore from backup
      gunzip -c /home/ubuntu/backups/investor_website/latest/database_backup.sql.gz | \
      docker-compose exec -T postgres psql -U investor_user -d investor_db
   ```

2. **Container Recovery:**
   ```bash
   # Restore from snapshot
   pct rollback 105 pre-production-20240101

# Or recreate container
pct destroy 105
# … recreate using original commands
```

Maintenance Procedures

Regular Maintenance Tasks

Daily:
- Monitor application logs
- Check service health
- Verify backup completion

Weekly:
- Update container packages
- Review security logs
- Test backup restoration

Monthly:
- Renew SSL certificates
- Update Docker images

- Performance optimization review
- Security audit

Update Procedures

```
# Update container OS
apt update && apt upgrade -y

# Update Docker images
docker-compose pull
docker-compose up -d

# Update Next.js application
cd /home/deploy/arcade_investor_site
git pull origin main
cd /home/deploy/proxmox_deployment/docker
docker-compose build nextjs_app
docker-compose up -d nextjs_app
```

Support and Resources

Useful Commands

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

# Follow all logs
docker-compose logs -f

# Access specific service
docker-compose exec nextjs_app bash
docker-compose exec postgres psql -U investor_user -d investor_db

# Restart specific service
docker-compose restart nginx

# View resource usage
docker stats

# Backup database manually
docker-compose exec postgres pg_dump -U investor_user investor_db > backup.sql
```

Configuration Files Reference

- **Docker Compose:** `/home/deploy/proxmox_deployment/docker/docker-compose.yml`
- **Nginx Config:** `/home/deploy/proxmox_deployment/nginx/nginx.conf`
- **Database Init:** `/home/deploy/proxmox_deployment/sql/init.sql`
- **Environment:** `/home/deploy/proxmox_deployment/docker/.env`
- **SSL Certificates:** `/home/deploy/proxmox_deployment/nginx/ssl/`

Getting Help

1. **Check Documentation:**
   - `PROXMOX_LXC_SETUP.md` - Container setup issues
   - `AUTH_SETUP.md` - Authentication problems
   - `README.md` - General overview

2. **Log Analysis:**

```bash
# Application logs
docker-compose logs nextjs_app

# Database logs
docker-compose logs postgres

# Nginx logs
docker-compose logs nginx

# System logs
journalctl -u docker
```

1. **Health Checks:**

```bash
# Service health
curl -k https://investor.local/api/health

# Database health
docker-compose exec postgres pg_isready

# Container health
pct status 105
```

Congratulations! Your gaming-themed investor website should now be fully deployed and operational on your Proxmox server. The system is configured for high availability, security, and scalability within your internal network.