Production Deployment Guide - Sports Bar TV Controller

Intel NUC13ANHi5 (i5-1340P) Deployment

Table of Contents

1. Hardware Overview
2. Pre-Migration Preparation
3. Installation Steps
4. Configuration
5. Data Migration
6. Testing & Verification
7. Performance Optimization
8. Troubleshooting
9. Rollback Procedures
10. Maintenance

Hardware Overview

Intel NUC13ANHi5 Specifications

- **CPU**: Intel Core i5-1340P (13th Gen)
- 12 cores (4 P-cores + 8 E-cores)
- 16 threads
- Base: 1.9 GHz, Turbo: up to 4.6 GHz
- 12MB Intel Smart Cache
- **RAM**: 16GB DDR4 (expandable to 64GB)
- **Storage**: 512GB NVMe SSD
- **GPU**: Intel Iris Xe Graphics (80 EUs)
- **Network**: Intel 2.5GbE LAN
- **Expected Performance**: 4-5x faster than current i5-7200U system

Performance Comparison

Component	Old System (i5-7200U)	New System (i5-1340P)	Improvement
CPU Cores	2 cores, 4 threads	12 cores, 16 threads	6x cores
Base Clock	2.5 GHz	1.9 GHz (P-cores)	-
Turbo Clock	3.1 GHz	4.6 GHz	48% faster
Cache	3MB	12MB	4x larger
GPU	Intel HD 620	Intel Iris Xe	2-3x faster
RAM	8-16GB	16GB (expandable)	Same/Better

Pre-Migration Preparation

1. Current System Backup Checklist

Before starting migration, create complete backups:

```
# On OLD system (135.131.39.26:223)
cd /home/ubuntu/Sports-Bar-TV-Controller

# 1. Backup PostgreSQL database
sudo -u postgres pg_dump sportsbar_tv > ~/backup-$(date +%Y%m%d)/database.sql

# 2. Backup environment variables
cp .env ~/backup-$(date +%Y%m%d)/.env.backup

# 3. Backup knowledge base
tar -czf ~/backup-$(date +%Y%m%d)/knowledge-base.tar.gz .ai-assistant/

# 4. Backup PM2 configuration
pm2 save
cp ~/.pm2/dump.pm2 ~/backup-$(date +%Y%m%d)/pm2-dump.json

# 5. Export Ollama models list
ollama list > ~/backup-$(date +%Y%m%d)/ollama-models.txt

# 6. Backup custom scripts
tar -czf ~/backup-$(date +%Y%m%d)/custom-scripts.tar.gz *.sh

# 7. Document current configuration
cat << EOF > ~/backup-$(date +%Y%m%d)/system-info.txt
Node.js: $(node --version)
npm: $(npm --version)
PM2: $(pm2 --version)
PostgreSQL: $(psql --version)
Ollama: $(ollama --version)
OS: $(lsb_release -d)
Kernel: $(uname -r)
EOF
```

2. Download Backup to Local Machine

```
# On your local machine
mkdir -p ~/sports-bar-backup-$(date +%Y%m%d)
scp -P 223 -r ubuntu@135.131.39.26:~/backup-$(date +%Y%m%d)/* ~/sports-bar-backup-$(da
te +%Y%m%d)/
```

3. Verify Backup Integrity

```
# Check backup files
ls -lh ~/sports-bar-backup-$(date +%Y%m%d)/
md5sum ~/sports-bar-backup-$(date +%Y%m%d)/*
```

Installation Steps

Step-by-Step Deployment Guide

Phase 1: Hardware Setup (30 minutes)

1. **Unbox and Connect NUC13ANHi5**
   - Connect power adapter
   - Connect ethernet cable

- Connect monitor, keyboard, mouse (for initial setup)
- Power on the system

2. **Install Ubuntu Server 22.04 LTS**
   - Download: https://ubuntu.com/download/server
   - Create bootable USB drive
   - Boot from USB and follow installation wizard
   - Configure:
     - Hostname: `sports-bar-nuc13`
     - Username: `ubuntu`
     - Enable OpenSSH server
     - Install security updates
3. **Initial System Configuration**
   ```bash
   # Update system
   sudo apt update && sudo apt upgrade -y

# Set timezone
sudo timedatectl set-timezone America/New_York # Adjust to your timezone

# Configure hostname
sudo hostnamectl set-hostname sports-bar-nuc13

# Reboot
sudo reboot
```

Phase 2: System Setup (45 minutes)

1. **Clone Repository**
   ```
   bash
      cd ~
      git clone https://github.com/dfultonthebar/Sports-Bar-TV-Controller.git
      cd Sports-Bar-TV-Controller
      git checkout production-deployment-nuc13
   ```

2. **Run System Setup Script**
   ```
   bash
      chmod +x scripts/*.sh
      ./scripts/system-setup.sh
   ```

What this script does:
- Installs Node.js 20.x LTS
- Installs PM2 process manager
- Installs PostgreSQL 15
- Installs Ollama
- Configures Intel Iris Xe graphics
- Sets up monitoring tools
- Configures firewall

Expected output:

```
[✓] System setup completed successfully!
   [!] IMPORTANT: Please reboot the system to apply kernel parameters for GPU optimization.
```

1. **Reboot System**
   ```
   bash
      sudo reboot
   ```

Phase 3: Ollama Setup (30 minutes)

1. **Configure and Optimize Ollama**
   ```
   bash
      cd ~/Sports-Bar-TV-Controller
      ./scripts/ollama-setup.sh
   ```

What this script does:
- Configures Ollama for Intel Iris Xe GPU
- Optimizes for 12-core CPU
- Sets memory limits for 16GB RAM
- Pulls required AI models (llama3.2:3b, qwen2.5:3b)
- Creates monitoring scripts

Expected output:

```
[✓] Ollama setup completed successfully!
   Ollama Configuration Summary:
     - Service: Running on 0.0.0.0:11434
     - Models: llama3.2:3b, qwen2.5:3b
     - CPU Threads: 10 (optimized for 12-core CPU)
     - Intel GPU: Enabled (Iris Xe)
     - Max VRAM: 4GB
```

1. **Verify Ollama Installation**
   ```bash
   # Check service status
   systemctl status ollama

# List installed models
ollama list

# Test model inference
ollama run llama3.2:3b “Hello, test response”
```

Phase 4: Application Deployment (30 minutes)

1. **Deploy Application**
   ```
   bash
      cd ~/Sports-Bar-TV-Controller
      ./scripts/app-deploy.sh
   ```

What this script does:
- Clones/updates repository to /opt/sports-bar-tv
- Installs Node.js dependencies
- Creates PostgreSQL database and user
- Generates .env configuration file
- Runs database migrations

- Builds Next.js application (optimized for 12 cores)
- Configures PM2 with 10 cluster instances
- Starts application

Expected output:

```
[✓] Application deployed successfully!
   Application URL: http://localhost:3000
```

1. **Verify Application Status**
   ```bash
   # Check PM2 status
   pm2 status

# View logs
pm2 logs sports-bar-tv –lines 50

# Test application
curl http://localhost:3000
```

Phase 5: Data Migration (45 minutes)

1. **Run Data Migration Script**
   ```
   bash
      cd ~/Sports-Bar-TV-Controller
      ./scripts/data-migration.sh
   ```

What this script does:
- Connects to old system via SSH
- Backs up PostgreSQL database
- Backs up environment variables
- Backs up knowledge base
- Backs up PM2 configuration
- Restores database to new system
- Restores knowledge base
- Pulls Ollama models from old system
- Restarts application

You will be prompted for:
- Confirmation to proceed
- SSH password for old system (6809233DjD$$$)

Expected output:

```
[✓] Migration completed successfully!
   Backup location: ~/migration-backup-YYYYMMDD-HHMMSS
```

1. **Verify Migration**
   ```bash
   # Check database
   sudo -u postgres psql -d sportsbar_tv -c “SELECT COUNT(*) FROM users;”

# Check knowledge base
ls -lh /opt/sports-bar-tv/.ai-assistant/
   ```

# Check application
pm2 logs sports-bar-tv –lines 20

Phase 6: Performance Setup (20 minutes)

1. **Configure Performance Monitoring**
   ```bash
   cd ~/Sports-Bar-TV-Controller
   ./scripts/performance-setup.sh
   ```

What this script does:
- Optimizes PostgreSQL for 12-core CPU and 16GB RAM
- Creates performance monitoring scripts
- Sets up automated performance reports (hourly)
- Configures weekly system optimization
- Installs benchmarking tools

1. **Run Initial Performance Check**
   ```bash
   ~/monitor-performance.sh
   ```

Configuration

Environment Variables

Edit `/opt/sports-bar-tv/.env`:

```
# Database Configuration
DATABASE_URL="postgresql://sportsbar:YOUR_SECURE_PASSWORD@localhost:5432/sportsbar_tv"

# Application Configuration
NODE_ENV=production
PORT=3000
NEXT_PUBLIC_API_URL=http://YOUR_SERVER_IP:3000

# Ollama Configuration
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_MODEL=llama3.2:3b

# Session Secret (already generated)
SESSION_SECRET=<auto-generated>

# External API Keys (configure as needed)
YOUTUBE_API_KEY=your_youtube_api_key
TWITCH_CLIENT_ID=your_twitch_client_id
TWITCH_CLIENT_SECRET=your_twitch_client_secret
```

After editing, restart the application:

```
pm2 restart sports-bar-tv
```

PostgreSQL Optimization

The performance-setup.sh script automatically configures PostgreSQL with these optimized settings:

```
# Memory Settings (optimized for 16GB RAM)
shared_buffers = 4GB                    # 25% of RAM
effective_cache_size = 12GB             # 75% of RAM
maintenance_work_mem = 1GB
work_mem = 64MB

# Parallel Query Settings (optimized for 12 cores)
max_worker_processes = 12
max_parallel_workers_per_gather = 6
max_parallel_workers = 12
max_parallel_maintenance_workers = 4

# Connection Settings
max_connections = 200

# SSD Optimization
random_page_cost = 1.1
effective_io_concurrency = 200
```

PM2 Cluster Configuration

The app-deploy.sh script configures PM2 with these settings:

```
{
  instances: 10,              // 10 instances for 12-core CPU
  exec_mode: 'cluster',       // Cluster mode for load balancing
  max_memory_restart: '1G',   // Auto-restart if memory exceeds 1GB
  autorestart: true,          // Auto-restart on crash
  watch: false,               // Disable file watching in production
  max_restarts: 10,           // Max restart attempts
  min_uptime: '10s'           // Minimum uptime before considering stable
}
```

Ollama Configuration

Optimized for Intel i5-1340P and Iris Xe:

```
OLLAMA_HOST=0.0.0.0:11434
OLLAMA_NUM_PARALLEL=4           # Parallel requests
OLLAMA_MAX_LOADED_MODELS=2      # Max models in memory
OLLAMA_INTEL_GPU=1              # Enable Intel GPU
OLLAMA_NUM_THREADS=10           # CPU threads (leaving 2 for system)
OLLAMA_MAX_VRAM=4096            # Max VRAM in MB
SYCL_CACHE_PERSISTENT=1         # Enable SYCL cache
BIGDL_LLM_XMX_DISABLED=1        # Disable XMX for iGPU
```

Data Migration

Manual Migration Steps (if script fails)

1. Database Migration

```
# On OLD system - Export database
sudo -u postgres pg_dump sportsbar_tv > /tmp/database.sql

# Transfer to NEW system
scp /tmp/database.sql ubuntu@NEW_SYSTEM_IP:/tmp/

# On NEW system - Import database
sudo -u postgres psql -d sportsbar_tv < /tmp/database.sql
```

2. Knowledge Base Migration

```
# On OLD system
tar -czf /tmp/knowledge-base.tar.gz -C /home/ubuntu/Sports-Bar-TV-Controller .ai-as-
sistant/

# Transfer to NEW system
scp /tmp/knowledge-base.tar.gz ubuntu@NEW_SYSTEM_IP:/tmp/

# On NEW system
tar -xzf /tmp/knowledge-base.tar.gz -C /opt/sports-bar-tv/
```

3. Environment Variables Migration

```
# On OLD system
cat /home/ubuntu/Sports-Bar-TV-Controller/.env

# Manually copy values to NEW system
nano /opt/sports-bar-tv/.env
```

4. Ollama Models Migration

```
# On OLD system - List models
ollama list

# On NEW system - Pull each model
ollama pull llama3.2:3b
ollama pull qwen2.5:3b
# ... pull other models as needed
```

Testing & Verification

Comprehensive Testing Checklist

1. System Health Checks

```
# CPU and Memory
htop

# Disk Space
df -h

# Network
ip addr show
ping -c 4 google.com

# Services Status
systemctl status postgresql
systemctl status ollama
pm2 status
```

2. Database Connectivity

```
# Test PostgreSQL connection
sudo -u postgres psql -d sportsbar_tv -c "SELECT version();"

# Check database size
sudo -u postgres psql -d sportsbar_tv -c "SELECT
pg_size_pretty(pg_database_size('sportsbar_tv'));"

# Verify tables
sudo -u postgres psql -d sportsbar_tv -c "\dt"

# Check data integrity
sudo -u postgres psql -d sportsbar_tv -c "SELECT COUNT(*) FROM users;"
```

3. Application Functionality Tests

```
# Test homepage
curl -I http://localhost:3000

# Test API endpoint
curl http://localhost:3000/api/health

# Test AI chat endpoint
curl -X POST http://localhost:3000/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "Hello, test message"}'
```

4. AI Chat Performance Tests

```
# Test Ollama directly
time ollama run llama3.2:3b "What is the capital of France?"

# Test through application
curl -X POST http://localhost:3000/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What sports are popular in bars?"}' \
  -w "\nTime: %{time_total}s\n"
```

5. Streaming Platform Integration Tests

Test each streaming platform:

- YouTube integration
- Twitch integration
- ESPN integration
- NFL Sunday Ticket integration

```
# Check API keys are configured
grep -E "YOUTUBE|TWITCH|ESPN" /opt/sports-bar-tv/.env

# Test streaming endpoints (adjust URLs as needed)
curl http://localhost:3000/api/streams/youtube
curl http://localhost:3000/api/streams/twitch
```

6. Performance Benchmarks

```
# Run system benchmark
~/benchmark-system.sh

# Monitor real-time performance
~/monitor-performance.sh

# Load test with Apache Bench (install if needed)
sudo apt install apache2-utils
ab -n 1000 -c 10 http://localhost:3000/
```

Expected Performance Metrics:

Metric	Target	Acceptable
Homepage Load Time	< 500ms	< 1s
API Response Time	< 200ms	< 500ms
AI Chat Response	< 3s	< 5s
Database Query	< 50ms	< 100ms
CPU Usage (idle)	< 10%	< 20%
Memory Usage	< 8GB	< 12GB

7. Browser Testing

Open in browser and test:

1. Homepage loads correctly
2. Navigation works
3. TV control interface responds
4. AI chat interface works
5. Streaming platform integrations work
6. No console errors

```
# Open browser to application
# If on local network: http://SERVER_IP:3000
# If using Nginx: http://SERVER_IP
```

Performance Optimization

CPU Optimization

The system is configured to utilize all 12 cores efficiently:

1. **PM2 Cluster Mode**: 10 instances (leaving 2 cores for system)
2. **PostgreSQL Parallel Queries**: Up to 6 workers per query
3. **Ollama Threading**: 10 threads for AI inference

Monitor CPU usage:

```
# Real-time monitoring
htop

# Per-process CPU usage
pm2 monit

# CPU statistics
mpstat -P ALL 1 5
```

Memory Optimization

16GB RAM allocation:

- PostgreSQL: 4GB shared buffers + 12GB effective cache
- Ollama: 4GB max VRAM
- PM2 instances: ~1GB each (10 instances = ~10GB)
- System: ~2GB reserved

Monitor memory:

```
# Memory usage
free -h

# Per-process memory
pm2 monit

# PostgreSQL memory
sudo -u postgres psql -c "SHOW shared_buffers;"
```

GPU Optimization (Intel Iris Xe)

Verify GPU is being utilized:

```
# Check GPU status
intel_gpu_top

# Verify GuC/HuC firmware
dmesg | grep -i guc
dmesg | grep -i huc

# Check Ollama GPU usage
journalctl -u ollama -f
```

If GPU is not being utilized:

1. Verify kernel parameters:
   `bash`
   `   cat /proc/cmdline | grep i915`

2. Check i915 module options:
   `bash`
   `   modinfo i915 | grep enable_guc`

3. Manually load firmware:
   `bash`
   `   sudo modprobe -r i915`
   `   sudo modprobe i915 enable_guc=3`

Network Optimization

For high-traffic scenarios:

```
# Increase network buffers
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.core.wmem_max=16777216

# Enable TCP BBR congestion control
sudo sysctl -w net.ipv4.tcp_congestion_control=bbr

# Make permanent
echo "net.core.rmem_max=16777216" | sudo tee -a /etc/sysctl.conf
echo "net.core.wmem_max=16777216" | sudo tee -a /etc/sysctl.conf
echo "net.ipv4.tcp_congestion_control=bbr" | sudo tee -a /etc/sysctl.conf
```

Storage Optimization

SSD optimization:

```
# Enable TRIM
sudo systemctl enable fstrim.timer
sudo systemctl start fstrim.timer

# Check TRIM status
sudo fstrim -v /
```

Troubleshooting

Common Issues and Solutions

Issue 1: Application Won't Start

Symptoms:
- PM2 shows app as "errored" or "stopped"
- Error logs show connection issues

Solutions:

1. Check PostgreSQL is running:
   ```
   bash
      sudo systemctl status postgresql
      sudo systemctl start postgresql
   ```

2. Verify database connection:
   ```
   bash
      sudo -u postgres psql -d sportsbar_tv -c "SELECT 1;"
   ```

3. Check environment variables:
   ```
   bash
      cat /opt/sports-bar-tv/.env
   ```

4. Review error logs:
   ```
   bash
      pm2 logs sports-bar-tv --err --lines 100
   ```

5. Restart application:
   ```
   bash
      pm2 restart sports-bar-tv
   ```

Issue 2: Ollama Not Responding

Symptoms:
- AI chat returns errors
- Ollama service not running

Solutions:

1. Check Ollama service:
   ```
   bash
      systemctl status ollama
      sudo systemctl restart ollama
   ```

2. Verify models are loaded:
   ```
   bash
      ollama list
   ```

3. Test Ollama directly:
   ```
   bash
      curl http://localhost:11434/api/tags
   ```

4. Check Ollama logs:
   ```
   bash
      sudo journalctl -u ollama -n 100
   ```

5. Re-pull models if needed:
   ```
   bash
      ollama pull llama3.2:3b
   ```

Issue 3: High CPU Usage

Symptoms:
- CPU usage consistently above 80%
- System feels sluggish

Solutions:

1. Check which process is consuming CPU:
   ```
   bash
      htop
      pm2 monit
   ```

2. Reduce PM2 instances if needed:
   ```
   bash
      pm2 scale sports-bar-tv 8  # Reduce from 10 to 8
   ```

3. Optimize PostgreSQL queries:
   ```
   bash
      sudo -u postgres psql -d sportsbar_tv -c "SELECT * FROM pg_stat_activity WHERE state =
   'active';"
   ```

4. Check for runaway processes:
   ```bash
   ps aux | sort -nrk 3,3 | head -n 10
   ```

Issue 4: Memory Issues

Symptoms:
- Out of memory errors
- System swapping heavily

Solutions:

1. Check memory usage:
   ```bash
   free -h
   vmstat 1 5
   ```

2. Identify memory-hungry processes:
   ```bash
   ps aux | sort -nrk 4,4 | head -n 10
   ```

3. Reduce PM2 memory limits:
   ```bash
   # Edit ecosystem.config.js
   nano /opt/sports-bar-tv/ecosystem.config.js
   # Change max_memory_restart to 800M
   pm2 restart sports-bar-tv
   ```

4. Reduce Ollama VRAM:
   ```bash
   sudo nano /etc/systemd/system/ollama.service.d/override.conf
   # Change OLLAMA_MAX_VRAM to 3072
   sudo systemctl daemon-reload
   sudo systemctl restart ollama
   ```

Issue 5: Database Connection Errors

Symptoms:
- “Connection refused” errors
- “Too many connections” errors

Solutions:

1. Check PostgreSQL status:
   ```bash
   sudo systemctl status postgresql
   ```

2. Check connection count:
   ```bash
   sudo -u postgres psql -c "SELECT count(*) FROM pg_stat_activity;"
   ```

3. Increase max_connections if needed:
   ```bash
   sudo nano /etc/postgresql/15/main/postgresql.conf
   ```

```
   # Increase max_connections to 300
   sudo systemctl restart postgresql
```

4. Check for connection leaks:
   ```bash
   sudo -u postgres psql -c "SELECT * FROM pg_stat_activity WHERE state = 'idle in transac-
   tion';"
   ```

Issue 6: Intel GPU Not Working

Symptoms:
- Ollama not using GPU
- Poor AI inference performance

Solutions:

1. Verify GPU is detected:
   ```bash
   lspci | grep VGA
   vainfo
   ```

2. Check kernel parameters:
   ```bash
   cat /proc/cmdline | grep i915
   ```

3. Verify GuC firmware:
   ```bash
   dmesg | grep -i guc
   ```

4. Reload i915 module:
   ```bash
   sudo modprobe -r i915
   sudo modprobe i915 enable_guc=3
   ```

5. If still not working, disable GPU and use CPU:
   ```bash
   sudo nano /etc/systemd/system/ollama.service.d/override.conf
   # Remove or comment out OLLAMA_INTEL_GPU=1
   sudo systemctl daemon-reload
   sudo systemctl restart ollama
   ```

Rollback Procedures

Emergency Rollback to Old System

If critical issues occur and you need to revert:

Step 1: Stop New System

```
# On NEW system
pm2 stop sports-bar-tv
sudo systemctl stop ollama
sudo systemctl stop postgresql
```

Step 2: Restart Old System

```
# On OLD system (135.131.39.26:223)
cd /home/ubuntu/Sports-Bar-TV-Controller
pm2 restart all
sudo systemctl start ollama
sudo systemctl start postgresql
```

Step 3: Verify Old System

```
# Check services
pm2 status
systemctl status ollama
systemctl status postgresql

# Test application
curl http://localhost:3000
```

Step 4: Update DNS/Routing

If you’ve updated DNS or routing to point to the new system, revert those changes to point back to the old system.

Partial Rollback (Database Only)

If only the database needs to be rolled back:

```
# On NEW system
sudo systemctl stop postgresql

# Restore from backup
sudo -u postgres psql -d sportsbar_tv < ~/migration-backup-YYYYMMDD-HHMMSS/
database.sql

# Restart
sudo systemctl start postgresql
pm2 restart sports-bar-tv
```

Rollback Checklist

- [ ] Stop services on new system
- [ ] Verify old system is operational
- [ ] Test old system functionality
- [ ] Update DNS/routing if needed
- [ ] Notify users of rollback
- [ ] Document issues encountered
- [ ] Plan for retry/fixes

Maintenance

Daily Maintenance

Automated (via cron):
- Hourly performance reports
- Log rotation

Manual checks:

```
# Quick health check
~/monitor-performance.sh

# Check PM2 status
pm2 status

# Check disk space
df -h
```

Weekly Maintenance

Automated (via cron - Sundays at 2 AM):
- System optimization
- Database VACUUM
- Log cleanup

Manual tasks:

```
# Review performance reports
ls -lh ~/performance-reports/

# Check for system updates
sudo apt update
sudo apt list --upgradable

# Review error logs
pm2 logs sports-bar-tv --err --lines 100
```

Monthly Maintenance

```
# Full system update
sudo apt update && sudo apt upgrade -y

# Database maintenance
sudo -u postgres psql -d sportsbar_tv -c "VACUUM FULL ANALYZE;"

# Check for Ollama updates
ollama --version
# Visit https://ollama.com for latest version

# Review and clean old backups
ls -lh ~/migration-backup-*/
# Delete backups older than 30 days
find ~ -name "migration-backup-*" -mtime +30 -exec rm -rf {} \;

# Review performance trends
cat ~/performance-reports/perf-report-*.txt | grep "CPU Usage"
```

Backup Strategy

Daily backups (automated):

```
# Add to crontab
crontab -e

# Add this line for daily 2 AM backups
0 2 * * * /opt/sports-bar-tv/scripts/backup-daily.sh
```

Create backup script:

```
cat << 'EOF' > /opt/sports-bar-tv/scripts/backup-daily.sh
#!/bin/bash
BACKUP_DIR="/backup/daily/$(date +%Y%m%d)"
mkdir -p $BACKUP_DIR

# Backup database
sudo -u postgres pg_dump sportsbar_tv | gzip > $BACKUP_DIR/database.sql.gz

# Backup knowledge base
tar -czf $BACKUP_DIR/knowledge-base.tar.gz -C /opt/sports-bar-tv .ai-assistant/

# Backup environment
cp /opt/sports-bar-tv/.env $BACKUP_DIR/.env.backup

# Clean old backups (keep 7 days)
find /backup/daily -mtime +7 -exec rm -rf {} \;
EOF

chmod +x /opt/sports-bar-tv/scripts/backup-daily.sh
```

Monitoring and Alerts

Set up email alerts for critical issues:

```
# Install mailutils
sudo apt install mailutils

# Create alert script
cat << 'EOF' > ~/alert-check.sh
#!/bin/bash
EMAIL="your-email@example.com"

# Check CPU usage
CPU=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100
- $1}')
if (( $(echo "$CPU > 90" | bc -l) )); then
    echo "High CPU usage: $CPU%" | mail -s "Alert: High CPU on NUC13" $EMAIL
fi

# Check memory usage
MEM=$(free | grep Mem | awk '{print ($3/$2) * 100.0}')
if (( $(echo "$MEM > 90" | bc -l) )); then
    echo "High memory usage: $MEM%" | mail -s "Alert: High Memory on NUC13" $EMAIL
fi

# Check disk space
DISK=$(df -h / | tail -1 | awk '{print $5}' | sed 's/%//')
if [ $DISK -gt 90 ]; then
    echo "High disk usage: $DISK%" | mail -s "Alert: High Disk Usage on NUC13" $EMAIL
fi

# Check PM2 status
if ! pm2 status | grep -q "online"; then
    echo "PM2 application not running" | mail -s "Alert: PM2 Down on NUC13" $EMAIL
fi
EOF

chmod +x ~/alert-check.sh

# Add to crontab (check every 15 minutes)
crontab -e
# Add: */15 * * * * ~/alert-check.sh
```

Security Updates

Enable automatic security updates:

```
sudo apt install unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades
```

Configure update settings:

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Performance Tuning Tips

Fine-Tuning for Your Workload

High Traffic Scenarios

If experiencing high concurrent users:

1. Increase PM2 instances:
   `bash`
   ```
   pm2 scale sports-bar-tv 12  # Use all 12 cores
   ```

2. Increase PostgreSQL connections:
   `bash`
   ```
   sudo nano /etc/postgresql/15/main/postgresql.conf
   # Set max_connections = 300
   sudo systemctl restart postgresql
   ```

3. Enable Nginx caching:
   `bash`
   ```
   sudo apt install nginx
   # Configure Nginx as reverse proxy with caching
   ```

AI-Heavy Workload

If AI chat is heavily used:

1. Increase Ollama parallel requests:
   `bash`
   ```
   sudo nano /etc/systemd/system/ollama.service.d/override.conf
   # Set OLLAMA_NUM_PARALLEL=6
   sudo systemctl daemon-reload
   sudo systemctl restart ollama
   ```

2. Use smaller, faster models:
   `bash`
   ```
   ollama pull llama3.2:1b  # Smaller, faster model
   ```

3. Implement response caching in application

Database-Heavy Workload

If database queries are slow:

1. Increase work_mem:
   `bash`
   ```
   sudo nano /etc/postgresql/15/main/postgresql.conf
   # Set work_mem = 128MB
   sudo systemctl restart postgresql
   ```

2. Add database indexes:
   `bash`
   ```
   sudo -u postgres psql -d sportsbar_tv
   # Analyze slow queries and add indexes
   ```

3. Enable query result caching in application

Security Considerations

Firewall Configuration

```
# Allow only necessary ports
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw allow 3000/tcp  # Application (if direct access needed)
sudo ufw enable
```

SSL/TLS Configuration

Set up SSL with Let's Encrypt:

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Get certificate
sudo certbot --nginx -d your-domain.com

# Auto-renewal is configured automatically
```

Database Security

```
# Change default PostgreSQL password
sudo -u postgres psql
ALTER USER sportsbar WITH PASSWORD 'new_secure_password';

# Update .env file
nano /opt/sports-bar-tv/.env
# Update DATABASE_URL with new password

# Restart application
pm2 restart sports-bar-tv
```

Regular Security Audits

```
# Check for security updates
sudo apt update
sudo apt list --upgradable | grep -i security

# Review open ports
sudo netstat -tulpn

# Check for failed login attempts
sudo grep "Failed password" /var/log/auth.log

# Review PM2 logs for suspicious activity
pm2 logs sports-bar-tv | grep -i error
```

Conclusion

This deployment guide provides comprehensive instructions for migrating the Sports Bar TV Controller to the Intel NUC13ANHi5 system. The new hardware offers significant performance improvements with its 12-core CPU, 16GB RAM, and Intel Iris Xe graphics.

Key Takeaways

1. **Performance**: Expect 4-5x performance improvement over the old system
2. **Scalability**: PM2 cluster mode with 10 instances handles high traffic
3. **Optimization**: PostgreSQL and Ollama are tuned for the hardware
4. **Monitoring**: Automated performance monitoring and alerts
5. **Reliability**: Automated backups and rollback procedures

Next Steps After Deployment

1. Monitor performance for first 24 hours
2. Fine-tune based on actual workload
3. Set up external monitoring (optional)
4. Configure SSL/TLS for production
5. Set up automated backups to external storage
6. Document any custom configurations

Support Resources

- **GitHub Repository**: https://github.com/dfultonthebar/Sports-Bar-TV-Controller
- **Deployment Branch**: production-deployment-nuc13
- **Performance Reports**: ~/performance-reports/
- **Backup Location**: ~/migration-backup-*/

Emergency Contacts

- **Old System**: 135.131.39.26:223 (keep online for 30 days as backup)
- **New System**: [Configure after deployment]

Document Version: 1.0
Last Updated: October 7, 2025
Target Hardware: Intel NUC13ANHi5 (i5-1340P)
Application: Sports Bar TV Controller
