

Update Script Documentation

Overview

The `update_from_github.sh` script has been significantly enhanced to provide a robust, safe, and intelligent update mechanism for the Sports Bar TV Controller application. This document explains the improvements and safety measures implemented.

Key Improvements

1. Automatic Package Manager Detection

The script now intelligently detects whether to use Yarn or npm:

- **Primary Detection:** Checks for `yarn.lock` file
- **Fallback Logic:** If Yarn is not installed but `yarn.lock` exists, attempts to install Yarn globally
- **Safe Fallback:** Falls back to npm if Yarn installation fails
- **npm Support:** Uses npm if `package-lock.json` is present

Why This Matters: Previously, the script always used npm, which could cause dependency conflicts in a Yarn project. This led to broken dependencies and application failures.

2. Smart Dependency Installation

Dependencies are only reinstalled when necessary:

```
# Before git pull: Calculate MD5 hashes of package.json and lock files
# After git pull: Compare hashes
# Only run install if files changed
```

Benefits:

- Saves time on updates that don't change dependencies
- Prevents breaking working dependencies unnecessarily
- Reduces risk of version conflicts
- Faster update process

Commands Used:

- Yarn: `yarn install --frozen-lockfile` (ensures exact versions from lock file)
- npm: `npm ci` (clean install from lock file)

3. Graceful Server Management

Stopping the Server

- Finds process on port 3000 using `lsof`
- Sends SIGTERM for graceful shutdown
- Waits up to 10 seconds for clean exit
- Force kills only if necessary (SIGKILL)
- Cleans up any orphaned npm/next processes

Starting the Server

- Uses `nohup` to run server in background
- Captures PID for monitoring
- Waits up to 30 seconds for server to respond
- Verifies server is actually serving requests
- Provides clear feedback on startup status

4. Enhanced Error Handling

```
set -e                # Exit immediately on error
set -o pipefail       # Catch errors in pipes
trap cleanup_on_error ERR # Run cleanup on any error
```

Error Recovery:

- All errors are logged with timestamps
- System state is preserved on failure
- Backup is always created before changes
- Clear error messages guide troubleshooting

5. Comprehensive Logging

All operations are logged to `update.log` with timestamps:

```
[2025-10-01 19:30:15] 🔄 Starting Sports Bar AI Assistant Update
[2025-10-01 19:30:16] 📦 Detected package manager: yarn
[2025-10-01 19:30:17] ✅ Configuration backed up to: /home/ubuntu/sports-bar-backups/c
onfig-backup-20251001-193017.tar.gz
```

Log Levels:

- `log()` : General information
- `log_success()` : Successful operations (✅)
- `log_warning()` : Non-critical issues (⚠️)
- `log_error()` : Critical failures (❌)

6. Safety Checks

Pre-Update Checks

- Verifies project directory exists
- Checks for uncommitted local changes
- Warns about potential conflicts

During Update

- Backs up all configuration before changes
- Preserves local files (gitignored)
- Only cleans temporary build artifacts
- Never touches user data or database

Post-Update Verification

- Verifies server responds to HTTP requests
- Checks build completed successfully
- Confirms all services are running

7. Backup Management

```
BACKUP_DIR="$HOME/sports-bar-backups"
BACKUP_FILE="$BACKUP_DIR/config-backup-$(date +%Y%m%d-%H%M%S).tar.gz"
```

What's Backed Up:

- All local configuration files (config/*.local.json)
- Environment variables (.env)
- Database (prisma/dev.db)
- Data files (data/*.json)
- Scene logs and Atlas configs

Backup Retention:

- Keeps last 7 backups automatically
- Older backups are automatically deleted
- Each backup is timestamped

Restore Process:

```
tar -xzf /home/ubuntu/sports-bar-backups/config-backup-YYYYMMDD-HHMMSS.tar.gz
```

Usage

Basic Update

```
./update_from_github.sh
```

The script will:

1. Detect your package manager (Yarn/npm)
2. Backup your configuration
3. Stop the running server gracefully
4. Pull latest changes from GitHub
5. Install dependencies (only if needed)
6. Update database schema
7. Build the application
8. Restart the server
9. Verify everything is working

Monitoring the Update

Watch the update in real-time:

```
tail -f update.log
```

Check server status:

```
tail -f server.log
```

Troubleshooting

If the update fails:

1. **Check the logs:**

```
bash
cat update.log
cat server.log
```

2. **Restore from backup** (if needed):

```
bash
cd /home/ubuntu/Sports-Bar-TV-Controller
tar -xzf /home/ubuntu/sports-bar-backups/config-backup-YYYYMMDD-HHMMSS.tar.gz
```

3. **Manual server restart:**

```
bash
npm start
```

4. **Check server is running:**

```
bash
curl http://localhost:3000
```

Safety Features Summary

Feature	Protection	Benefit
Package Manager Detection	Prevents npm/yarn conflicts	Maintains dependency integrity
Smart Dependency Install	Only updates when needed	Avoids breaking working system
Graceful Shutdown	Clean process termination	Prevents data corruption
Automatic Backups	Configuration preserved	Easy rollback if needed
Error Handling	Stops on first error	Prevents cascading failures
Comprehensive Logging	Full audit trail	Easy troubleshooting
Server Verification	Confirms app is working	Catches deployment issues
Lock File Usage	Exact version matching	Reproducible builds

What's Preserved During Updates

The following are **NEVER** modified or deleted:

- ✓ **Database:** `prisma/dev.db`
- All your configurations
- Device settings

- Input/output mappings
- Scenes and presets
- API credentials

✓ **Configuration Files:** `config/*.local.json`

- System settings
- Device inventory
- Sports team preferences

✓ **Environment Variables:** `.env`

- API keys
- Secrets
- Service credentials

✓ **Data Files:** `data/*.json`

- Streaming credentials
- Device subscriptions
- Custom configurations

✓ **User Uploads:** `uploads/`

- Layout PDFs
- Custom documents

Technical Details

Package Manager Selection Logic

```
if [ -f "yarn.lock" ]; then
    if command -v yarn &> /dev/null; then
        USE_YARN=true
    else
        # Try to install yarn
        npm install -g yarn
        if command -v yarn &> /dev/null; then
            USE_YARN=true
        else
            USE_NPM=true # Fallback
        fi
    fi
elif [ -f "package-lock.json" ]; then
    USE_NPM=true
fi
```

Dependency Change Detection

```
# Before pull
md5sum package.json > /tmp/pre_update_package_hash
md5sum yarn.lock > /tmp/pre_update_lock_hash

# After pull
md5sum package.json > /tmp/post_update_package_hash
md5sum yarn.lock > /tmp/post_update_lock_hash

# Compare
if [ "$(cat /tmp/pre_update_package_hash)" != "$(cat /tmp/
post_update_package_hash)" ]; then
    # Dependencies changed - need to install
fi
```

Process Management

```
# Find process on port
pid=$(lsof -ti:3000)

# Graceful shutdown
kill -TERM $pid

# Wait for exit (max 10 seconds)
for i in {1..10}; do
    if ! kill -0 $pid 2>/dev/null; then
        break
    fi
    sleep 1
done

# Force kill if still running
if kill -0 $pid 2>/dev/null; then
    kill -9 $pid
fi
```

Best Practices

1. **Run updates during low-traffic periods** to minimize user impact
2. **Check logs after each update** to verify success
3. **Keep backups** - they're automatically managed but good to verify
4. **Test after updates** - the script verifies the server starts, but manual testing is recommended
5. **Monitor the first few requests** after an update to catch any issues early

Comparison: Old vs New Script

Aspect	Old Script	New Script
Package Manager	Always npm	Auto-detects Yarn/npm
Dependency Install	Always runs	Only when files change
Server Stop	Basic pkill	Graceful with timeout
Server Start	Simple background	Verified startup
Error Handling	Basic	Comprehensive with traps
Logging	Console only	File + console with timestamps
Backup	Basic	Managed with retention
Safety Checks	Minimal	Extensive

Future Enhancements

Potential improvements for future versions:

- ☐ Pre-update health check
- ☐ Rollback command
- ☐ Update notifications
- ☐ Scheduled updates
- ☐ Update history tracking
- ☐ Performance metrics
- ☐ Email notifications on failure

Support





If you encounter issues:

1. Check `update.log` for detailed error messages
2. Check `server.log` for application errors
3. Verify your configuration files are intact
4. Restore from backup if needed
5. Contact support with log files

Version History

Version 2.0 (Current)

- ☒ Automatic Yarn/npm detection
- ☒ Smart dependency installation

-  Graceful server management
-  Enhanced error handling
-  Comprehensive logging
-  Safety checks and verification

Version 1.0 (Original)

- Basic git pull
- npm install (always)
- Simple server restart
- Basic backup

Last Updated: October 1, 2025

Script Version: 2.0

Maintainer: Sports Bar TV Controller Team