Deployment Workflow Guide

Deployment Workflow Guide

Overview

Professional Music Teachers of New Mexico uses GitHub Actions to automatically deploy changes to the production and test environments. This guide explains how the deployment process works and how to use it effectively.

Current Deployment Status

Latest Version Information:

- **PMTNM Version**: {{api:version|version.pmtnm_version}}
- **Version Date**: {{api:version|version.pmtnm_version_date}}
- **Description**: {{api:version|version.pmtnm_version_desc}}
- **Site Version**: {{api:version|version.site_version}}
- **Git Commit**: {{api:version|version.git_commit_hash}}
- **Branch**: {{api:version|git.branch}}
- **Commit Date**: {{api:version|git.commit_date_formatted}}
- **Commit Message**: {{api:version|git.commit_message}}

Full API Response:

{{api:version}}

How It Works

Automatic Deployment

The deployment workflow is triggered automatically when code is pushed to specific branches:

- `test` **branch** → Deploys to `test.pmtnm.org`

- **`main` branch** → Deploys to `pmtnm.org` (production)

Deployment Process

The GitHub Actions workflow (`.github/workflows/deploy.yml`) performs the following steps:

1. **Environment Detection**
   - Determines whether this is a test or production deployment based on the branch
   - Sets appropriate FTP credentials and target paths
2. **Git Commit Information**
   - Captures the current commit hash, message, date, and branch
   - Stores this information in `data/.git_commit_hash.txt` , `data/.git_commit_message.txt` , and `data/.git_commit_info.json`
   - These files are always included in deployments to track what version is running
3. **Change Detection**
   - Compares the current commit with the previous commit (HEAD~1)
   - Identifies files that were added, changed, or deleted
   - Filters out excluded patterns (`.git` , `.github` , `node_modules` , `README.md`)
4. **File Upload**
   - Creates necessary remote directories
   - Uploads only changed files via FTP using `lftp`
   - Tracks successful uploads and skipped files
5. **File Deletion**
   - Removes files that were deleted from the repository
   - Handles deletion failures gracefully (files might not exist on server)
6. **Deployment Summary**
   - Provides a comprehensive summary of uploaded and deleted files

- Shows environment, branch, and target URL

Deployment Environments

Test Environment (`test.pmtnm.org`)

- **Branch**: `test`
- **Purpose**: Testing changes before production
- **FTP Credentials**: Stored in GitHub Secrets (`TEST_FTP_*`)

Production Environment (`pmtnm.org`)

- **Branch**: `main`
- **Purpose**: Live production site
- **FTP Credentials**: Stored in GitHub Secrets (`FTP_*`)

Version Tracking

Version Information

The deployment system tracks multiple types of version information:

1. **PMTNM Version** (`includes/version.php`)
   - Manual version number (e.g., `2025.12.17.01`)
   - Updated when significant changes are deployed
   - Includes version date and description

2. **Site Version** (Automatic)
   - Hash based on code file modification times
   - Automatically updates when any code file (PHP, CSS, JS) is modified
   - Format: `YYYY-MM-DD-hash` (e.g., `2025-12-17-a3f2b1c4`)

3. **Git Commit Hash**
   - Short hash (7 characters) from the deployed commit
   - Full hash (40 characters) for complete identification

- Commit message and date for reference

Accessing Version Information

Via API

The version information is available via the API endpoint:

```
GET /api/version.php
```

Example Response:

```
{
    "status": "ok",
    "timestamp": "2025-12-17T00:31:28+00:00",
    "environment": "pmtnm.org",
    "version": {
        "site_version": "2025-12-17-15bfbba3",
        "git_commit_hash": "abbf329",
        "git_commit_hash_full": "abbf3292d23ec11496fc25b281d7c11f5529e94f",
        "pmtnm_version": "2025.12.17.01",
        "pmtnm_version_date": "2025-12-17",
        "pmtnm_version_desc": "Enhanced click tracking and logging system"
    },
    "git": {
        "available": true,
        "branch": "main",
        "commit_date": "2025-12-16 17:27:27 -0700",
        "commit_message": "Deploy latest changes"
    }
}
```

Via Web Interface

Visit `/software_version.php` to see a formatted display of all version information.

Deployment Workflow Details

Step-by-Step Process

1. Code Push

```
git push origin test    # Deploys to test environment
git push origin main   # Deploys to production
```

2. GitHub Actions Trigger

- Workflow automatically starts when code is pushed
- Runs on `ubuntu-latest` runner

3. Change Detection

The workflow compares files between commits:

- **Added/Modified**: Files in `git diff --name-only HEAD~1 HEAD --diff-filter=ACM`
- **Deleted**: Files in `git diff --name-only HEAD~1 HEAD --diff-filter=D`
- **First Commit**: If no previous commit exists, all files are deployed

4. File Filtering

Excluded patterns:

- `.git/**`
- `.github/**`
- `node_modules/**`
- `README.md`

5. FTP Upload

- Uses `lftp` for reliable FTP transfers
- Creates remote directories as needed
- Uploads files preserving directory structure
- Tracks upload success/failure

6. File Deletion

- Removes deleted files from server
- Handles cases where files don't exist gracefully

7. Summary Report

- Lists all uploaded files

- Lists all deleted files
- Shows environment and target URL
- Displays any skipped files

Skipping Deployment

If no files are changed or deleted, the workflow automatically skips the deployment step to save time and resources.

Best Practices

Before Deploying to Production

1. **Test on Test Environment First**
   - Always push to `test` branch first
   - Verify changes work correctly on `test.pmtnm.org`
   - Check for any errors or issues

2. **Review Changes**
   - Use `git diff` to review what will be deployed
   - Ensure no sensitive data is included
   - Verify file paths are correct

3. **Update Version Number**
   - Update `PMTNM_VERSION` in `includes/version.php` for significant changes
   - Update version date and description

4. **Commit Messages**
   - Write clear, descriptive commit messages
   - These appear in the deployment summary and version API

Deployment Monitoring

1. **Check GitHub Actions**
   - View workflow runs in the GitHub Actions tab
   - Monitor for any failures or warnings
2. **Verify Deployment**
   - Check the target URL after deployment
   - Use `/api/version.php` to verify the deployed version
   - Test critical functionality
3. **Review Deployment Summary**
   - Check the workflow logs for the deployment summary
   - Verify all expected files were uploaded
   - Confirm deleted files were removed

Troubleshooting

Deployment Failures

FTP Connection Issues

- Verify FTP credentials in GitHub Secrets
- Check FTP server availability
- Ensure firewall allows connections

File Upload Failures

- Check file permissions
- Verify remote directory structure
- Review workflow logs for specific errors

Missing Files

- Verify files are not in `.gitignore`
- Check file paths are correct

- Ensure files are committed to the repository

Version Information Issues

Version API Not Working

- Check `includes/version.php` is accessible
- Verify `data/` directory is writable (for caching)
- Check PHP error logs

Git Commit Hash Missing

- Verify `data/.git_commit_hash.txt` exists
- Check GitHub Actions workflow updated git files
- Ensure `.git` directory or stored files are available

Related Documentation

- Version API Documentation
- Version Hash Explanation
- GitHub Actions Workflow File

Support

For issues or questions about the deployment process, please use the contact form to reach the administrators.

Last Updated: {{api:version|version.pmtnm_version_date}} | Current Version: {{api:version|version.pmtnm_version}}