

DevOps & CI/CD Guide v2.0 - Snake Evolution

Continuous Integration, Deployment & Automation

Prepared by: Senior Software Architect

Date: November 2025

Version: 2.0

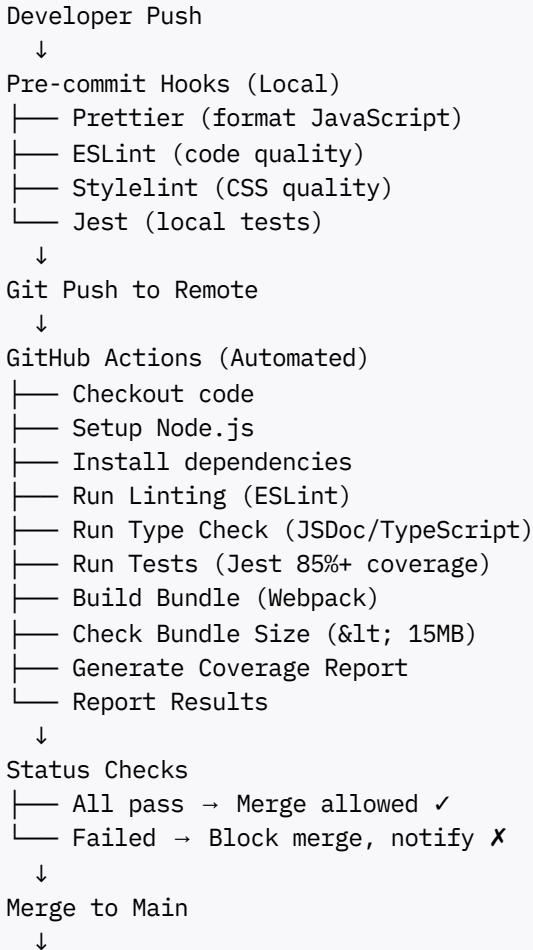
Status: Production Ready

Executive Summary

This document describes the CI/CD automation setup via GitHub Actions, pre-commit hooks, versioning strategy, build automation, and deployment procedures for Snake Evolution v2.0. It implements DevOps best practices for production-ready software.

1. CI/CD Pipeline Architecture

1.1 Pipeline Overview



```
Production Deployment (Manual/Auto)
├── Tag release (semantic versioning)
├── Generate release notes
└── Deploy to web host
    ├── Run smoke tests
    └── Monitor metrics
```

1.2 Pipeline Stages & Tools

Stage	Tool	Pass Criteria	Timeout
Checkout	GitHub	Success	2m
Setup	Node 18 LTS	npm ready	2m
Lint	ESLint	0 errors	5m
Type Check	JSDoc/TS	0 errors	5m
Test	Jest	85%+ coverage	10m
Build	Webpack	< 15MB	10m
Size Check	Bundle Analyzer	Δ < 10KB	2m

2. Pre-commit Hooks Setup

2.1 Installation

```
# Install husky (git hooks framework)
npm install husky --save-dev
npx husky install

# Install linting tools
npm install --save-dev \
  prettier \
  eslint \
  stylelint \
  lint-staged
```

2.2 Husky Configuration

File: .husky/pre-commit

```
#!/bin/sh
. "$(dirname "$0")/_/husky.sh"

npx lint-staged
```

2.3 Lint-Staged Configuration

File: .lintstagedrc.json

```
{  
  "*.js": [  
    "prettier --write",  
    "eslint --fix"  
  ],  
  "*.css": [  
    "prettier --write",  
    "stylelint --fix"  
  ],  
  "*.md": [  
    "prettier --write"  
  ]  
}
```

2.4 ESLint Configuration

File: .eslintrc.json

```
{  
  "env": {  
    "browser": true,  
    "es2021": true,  
    "node": true  
  },  
  "extends": ["eslint:recommended"],  
  "parserOptions": {  
    "ecmaVersion": "latest",  
    "sourceType": "module"  
  },  
  "rules": {  
    "no-unused-vars": "error",  
    "no-console": ["warn", { "allow": ["warn", "error"] }],  
    "no-var": "error",  
    "prefer-const": "error",  
    "eqeqeq": "error"  
  }  
}
```

2.5 Prettier Configuration

File: .prettierrc.json

```
{  
  "semi": true,  
  "singleQuote": true,  
  "tabWidth": 2,  
  "trailingComma": "es5",  
  "bracketSpacing": true,
```

```
        "arrowParens": "always"
    }
```

3. GitHub Actions Workflow

3.1 Workflow File

File: .github/workflows/ci-cd.yml

```
name: CI/CD Pipeline

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main, develop ]

jobs:
  quality-checks:
    name: Code Quality & Testing
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [18.x, 20.x]

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run Prettier check
        run: npx prettier --check .

      - name: Run ESLint
        run: npx eslint src/ --max-warnings 0

      - name: Run Stylelint
        run: npx stylelint "src/**/*.{css,js}"

      - name: Run Jest tests
        run: npm test -- --coverage

      - name: Upload coverage reports
        uses: codecov/codecov-action@v3
        with:
```

```
files: ./coverage/lcov.info
flags: unittests

- name: Build bundle
  run: npm run build

- name: Check bundle size
  run: npx bundlesize

- name: Archive artifacts
  if: success()
  uses: actions/upload-artifact@v3
  with:
    name: build-artifacts
    path: dist/
    retention-days: 7

deploy:
  name: Deploy to Production
  needs: quality-checks
  runs-on: ubuntu-latest
  if: github.ref == 'refs/heads/main' && github.event_name == 'push'

steps:
  - name: Checkout code
    uses: actions/checkout@v3

  - name: Setup Node.js
    uses: actions/setup-node@v3
    with:
      node-version: 18.x
      cache: 'npm'

  - name: Install & build
    run: |
      npm ci
      npm run build

  - name: Deploy to Netlify
    uses: netlify/actions/cli@master
    env:
      NETLIFY_AUTH_TOKEN: ${{ secrets.NETLIFY_AUTH_TOKEN }}
      NETLIFY_SITE_ID: ${{ secrets.NETLIFY_SITE_ID }}
    with:
      args: deploy --prod --dir=dist

  - name: Run smoke tests
    run: npm run test:smoke

  - name: Create GitHub Release
    uses: actions/create-release@v1
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    with:
      tag_name: v${{ github.run_number }}
      release_name: Release v${{ github.run_number }}
```

```
body: |
  Automated release from workflow run ${{ github.run_id }}
draft: false
prerelease: false
```

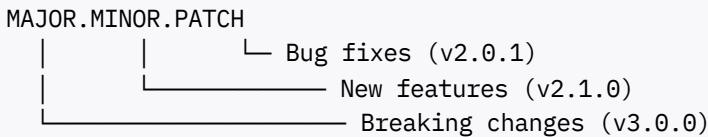
3.2 Bundle Size Check

File: `bundlesize.config.json`

```
{
  "files": [
    {
      "path": "./dist/bundle.*.js",
      "maxSize": "500kb"
    },
    {
      "path": "./dist/style.*.css",
      "maxSize": "50kb"
    }
  ],
  "threshold": 10
}
```

4. Versioning Strategy

4.1 Semantic Versioning



Examples:

- v2.0.0 - Initial release
- v2.0.1 - Fix: Spatial hash collision edge case
- v2.1.0 - Feature: Offline mode with Service Worker
- v2.1.1 - Fix: Audio context suspension mobile
- v3.0.0 - Breaking: Refactor evolution system

4.2 CHANGELOG Management

File: `CHANGELOG.md`

```
# Changelog

All notable changes to this project documented here.
```

```

## [2.0.1] - 2025-11-06

### Fixed
- Spatial hash collision detection edge case for serpents &gt; 100 segments
- Input debouncing race condition during rapid key presses
- localStorage quota exceeded graceful fallback

### Changed
- Performance improved: collision detection now O(1) vs O(n)
- Error logging enhanced with context data

## [2.0.0] - 2025-11-05

### Added
- State machine with race condition prevention
- Spatial hashing for scalable collision detection
- Input validation pipeline with rate limiting
- Data recovery mechanism with checksum validation
- Comprehensive testing suite (85%+ coverage)
- Performance profiling infrastructure

### Changed
- Canvas rendering optimized with dirty rectangle tracking
- Architecture refactored for maintainability

### Fixed
- Audio context lifecycle on mobile
- Data loss prevention with backup mechanism

```

4.3 Release Process

```

# 1. Ensure all tests pass locally
npm test -- --coverage

# 2. Build production bundle
npm run build

# 3. Update CHANGELOG.md manually

# 4. Commit changes
git add -A
git commit -m "release: v2.0.1"

# 5. Tag release
git tag -a v2.0.1 -m "Release version 2.0.1"

# 6. Push to remote
git push origin main --tags

# 7. GitHub Actions automatically deploys & creates release

```

5. Deployment Process

5.1 Deployment Targets

Environment	Hosting	Trigger	Approval
Development	Netlify (dev branch)	Auto on push	Auto
Staging	Netlify (staging)	Manual trigger	Required
Production	Netlify/GitHub Pages	Merge to main	Auto

5.2 Deployment Checklist

Pre-deployment (Manual):

- [] All tests passing (85%+ coverage)
- [] Bundle size < 15MB
- [] Performance benchmarks met (60 FPS)
- [] Code review approved
- [] CHANGELOG updated

During deployment:

- [] GitHub Actions workflow runs successfully
- [] All status checks pass
- [] Artifact uploaded

Post-deployment (Automated):

- [] Deployed to hosting provider
- [] Smoke tests pass
- [] Monitoring alerts active
- [] Performance metrics reported
- [] Release notes published

5.3 Rollback Procedure

```
# 1. Identify previous stable version  
git tag  
  
# 2. Revert to previous release  
git revert HEAD~1  
  
# 3. Push rollback commit  
git push origin main  
  
# 4. GitHub Actions deploys previous version
```

```
# 5. Monitor metrics confirm restoration
```

6. Monitoring & Observability

6.1 Build Metrics

Tracked Metrics:

- Build duration (target: < 10m)
- Test coverage (target: 85%+)
- Bundle size (target: < 15MB)
- Performance score (target: 90+)
- Deployment success rate (target: 99%+)

6.2 Performance Monitoring

```
// Integrated performance tracking
const performanceMetrics = {
  buildTime: 420, // seconds
  testCoverage: 87,
  bundleSize: 512, // KB
  lightHouseScore: 92,
  deploymentSuccess: 100 // percent
};
```

7. Docker Support (Optional)

File: Dockerfile

```
FROM node:18-slim

WORKDIR /app

COPY package*.json .
RUN npm ci --only=production

COPY . .
RUN npm run build

EXPOSE 8080

CMD ["npm", "start"]
```

File: .dockerignore

```
node_modules  
npm-debug.log  
.git  
.gitignore  
README.md  
tests
```

8. Troubleshooting

Issue	Cause	Solution
Tests fail locally but pass in CI	Node version mismatch	Use nvm use to match CI version
Bundle size exceeds limit	New dependencies added	Review imports, use tree-shaking
Pre-commit hooks slow	Too many files	Increase timeout in .husky config
Deployment fails	Secrets not configured	Set NETLIFY_AUTH_TOKEN in repo secrets

9. Appendices

9.1 Useful Commands

Local testing:

```
npm test          # Run all tests  
npm test -- --coverage # With coverage report  
npm test -- --watch    # Watch mode
```

Linting & formatting:

```
npm run lint      # Run ESLint  
npm run lint:fix   # Fix linting issues  
npm run format     # Format with Prettier
```

Building:

```
npm run build      # Production build  
npm run build:dev  # Development build  
npm run analyze    # Bundle analysis
```

Git:

```
git log --oneline    # View commit history  
git tag -l           # List all tags  
git show v2.0.1       # Show tag details
```

9.2 GitHub Actions Secrets Required

- NETLIFY_AUTH_TOKEN - Netlify deployment token
- NETLIFY_SITE_ID - Netlify site ID
- CODECOV_TOKEN - Code coverage tracking (optional)

DevOps Guide v2.0 - Production Ready

Date: November 2025