Phase 3 Usage Examples

Complete guide to using Solo Git's auto-merge workflow and CI orchestration

---

Table of Contents

Quick Start

The fastest way to promote code with Phase 3:

```
# Create workpad
sologit pad create --repo my-app --title "add-feature"

# Make changes
sologit pad apply-patch <pad-id> feature.patch

# Auto-merge (test + promote in one command)
sologit pad auto-merge <pad-id> --target fast
```

If tests pass, your code is instantly on trunk. If not, you get actionable feedback.

Basic Auto-Merge Workflow

Scenario: Adding a New Feature

Let's walk through adding a login feature:

Step 1: Create Workpad

```
$ sologit pad create --repo my-app --title "add-login-feature"
✅ Created workpad: pad_x7y6z5w4
```

Step 2: Develop Your Feature

Make your changes and create a patch:

```
$ git diff > login-feature.patch
```

Apply to workpad:

```
$ sologit pad apply-patch pad_x7y6z5w4 login-feature.patch
✅ Patch applied: checkpoint t1
```

Step 3: Run Auto-Merge

```
$ sologit pad auto-merge pad_x7y6z5w4 --target fast
```

Output:

```
🚀 Starting auto-merge workflow for: add-login-feature
   Target: fast
   Auto-promote: True

🧪 Running 5 tests...
   Tests completed in 3.2s

📊 Analyzing test results...
   Status: GREEN
   Passed: 5/5

🚦 Evaluating promotion gate...
   ✅ All tests passed (5/5)
   ✅ Can fast-forward to trunk
   📊 Change size: 8 files, ~156 lines
   🎉 All checks passed - ready to promote!

🚀 Auto-promoting to trunk...
   ✅ Promoted to trunk: a7b8c9d0

✅ SUCCESS
   Successfully promoted to trunk

📌 Commit: a7b8c9d0
```

Done! Your feature is now on trunk.

Working with Failed Tests

Scenario: Test Failure During Auto-Merge

```
$ sologit pad auto-merge pad_abc123 --target fast
```

Output:

```
🚀 Starting auto-merge workflow for: update-api

🧪 Running 5 tests...
   Tests completed in 2.8s

📊 Analyzing test results...
   Status: RED
   Passed: 4/5
   Failed: 1

   Failure Patterns:
     • IMPORT_ERROR: No module named 'requests'

   Suggested Actions:
     • 📦 Check missing dependencies - run 'pip install' for required packages
     • 🔍 Verify import paths and module names
     • 📝 Estimated fix complexity: LOW

🚦 Evaluating promotion gate...
   ❌ Tests failed: 1 failed, 0 timeout, 0 error

❌ FAILED
   Cannot promote - promotion gate rejected

   Fix the issues and try again:
   1. Address test failures
   2. Re-run tests: sologit test run <pad-id>
   3. Try again: sologit pad auto-merge <pad-id>
```

Fix and Retry

```
# 1. Fix the issue (add missing dependency)
$ echo "requests==2.31.0" >> requirements.txt
$ git add requirements.txt
$ git commit -m "Add missing dependency"

# 2. Apply fix to workpad
$ git format-patch HEAD~1 --stdout > fix.patch
$ sologit pad apply-patch pad_abc123 fix.patch

# 3. Try auto-merge again
$ sologit pad auto-merge pad_abc123 --target fast
✅ SUCCESS - Promoted to trunk: def456
```

Configuring Promotion Rules

Built-in Rule Sets

Solo Git supports flexible promotion policies. Configure in `~/.sologit/config.yaml`:

1. Production Policy (Strict)

```yaml
promotion:
  require_tests: true
  require_all_tests_pass: true
  require_fast_forward: true
  max_files_changed: 50
  max_lines_changed: 500
  allow_merge_conflicts: false
```

Use case: Production repositories where stability is critical.

Behavior:
- All tests must pass
- Maximum 50 files changed (large changes trigger manual review)
- Maximum 500 lines changed
- No merge conflicts allowed

2. Development Policy (Balanced)

```yaml
promotion:
  require_tests: true
  require_all_tests_pass: true
  require_fast_forward: true
  max_files_changed: null  # No limit
  max_lines_changed: null  # No limit
  allow_merge_conflicts: false
```

Use case: Active development where iteration speed matters.

Behavior:
- All tests must pass
- No size limits
- Fast-forward required (keep history clean)

3. Experimental Policy (Relaxed)

```yaml
promotion:
  require_tests: false  # Optional
  require_all_tests_pass: false
  require_fast_forward: true
  max_files_changed: null
  max_lines_changed: null
  allow_merge_conflicts: false
```

Use case: Rapid prototyping or spike work.

Behavior:
- Tests optional
- Can promote even with test failures (not recommended)
- Use for throw-away branches or experiments

Per-Workpad Rules

Override rules for specific workpads:

```
$ sologit pad create --repo my-app --title "hotfix" \
    --rules '{"require_tests": false}'
```

CI Smoke Tests

After-Promotion Validation

Phase 3 runs smoke tests **after** promotion to catch integration issues:

Configure Smoke Tests

In `evogit.yaml`:

```yaml
tests:
  fast: ["pytest tests/unit/"]

  smoke:
    - name: "integration"
      cmd: "pytest tests/integration/"
      timeout: 120

    - name: "e2e"
      cmd: "pytest tests/e2e/"
      timeout: 300

    - name: "performance"
      cmd: "./scripts/perf_test.sh"
      timeout: 180
```

Workflow

```
1. Developer auto-merges
   ↓
2. Fast tests pass → Promote to trunk
   ↓
3. CI smoke tests run in background
   ↓
4. If smoke tests fail → Auto-rollback
   ↓
5. Workpad recreated for fixes
```

Manual Smoke Test Run

```
$ sologit ci smoke my-app --commit abc123
```

Output:

```
🔬 Starting smoke tests for commit abc123...

Running 3 smoke tests...

✅ SUCCESS
   All smoke tests passed

Repository: my-app
Commit: abc123
Duration: 145.3s

Tests: 3
  ✅ Passed: 3
```

Rollback and Recovery

Automatic Rollback

If CI smoke tests fail after promotion, Solo Git automatically rolls back:

```
❌ CI Smoke Test Failed
   Test: integration-test
   Error: Connection to database failed

⏪ Initiating automatic rollback...
   ✅ Reverted commit: abc123
   ✅ Created fix workpad: pad_fix_abc123

📝 To fix the issue:
   1. Work in workpad: pad_fix_abc123
   2. Fix the failing tests
   3. Run: sologit test run pad_fix_abc123
   4. Try again: sologit pad auto-merge pad_fix_abc123
```

Manual Rollback

```
# Rollback specific commit
$ sologit ci rollback my-app --commit abc123
```

Options:

```
# Rollback without recreating workpad
$ sologit ci rollback my-app --commit abc123 --no-recreate-pad

# Rollback and specify workpad title
$ sologit ci rollback my-app --commit abc123 --pad-title "hotfix-database"
```

Advanced Scenarios

Scenario 1: Large Refactoring

Challenge: Refactoring touches 100+ files.

Solution: Use manual review workflow:

```
# 1. Create workpad
$ sologit pad create --repo my-app --title "refactor-auth"

# 2. Apply changes
$ sologit pad apply-patch <pad-id> refactor.patch

# 3. Run tests (without auto-promote)
$ sologit pad auto-merge <pad-id> --no-auto-promote
```

Output:

```
🚦 Evaluation Result: MANUAL_REVIEW

   ⚠️ Large change: 125 files (limit: 50)
   ⚠️ Large change: 3200 lines (limit: 500)

   Tests passed, but manual review recommended

   To promote manually:
   $ sologit pad promote <pad-id>
```

Scenario 2: Multiple Workpads

Challenge: Working on multiple features simultaneously.

Solution: Each workpad is isolated:

```
# Feature A
$ sologit pad create --repo my-app --title "feature-a"
$ sologit pad auto-merge <pad-a-id>

# Feature B (independent)
$ sologit pad create --repo my-app --title "feature-b"
$ sologit pad auto-merge <pad-b-id>
```

Both can be auto-merged independently. If both pass, both merge to trunk.

Scenario 3: Failed CI After Multiple Promotions

Challenge: Multiple commits promoted, CI fails.

Current behavior: Only the last commit is rolled back.

Workaround:

```
# Check CI status
$ sologit ci status my-app

# Manual rollback of multiple commits
$ sologit ci rollback my-app --commit abc123
$ sologit ci rollback my-app --commit def456
```

Future enhancement: Batch rollback of related commits.

Scenario 4: Dry Run Evaluation

Challenge: Want to check if workpad will pass without promoting.

Solution:

```
$ sologit pad evaluate <pad-id>
```

Output:

```
🚦 PROMOTION GATE EVALUATION

✅ APPROVED - Ready to promote

Reasons:
  ✅ All tests passed (8/8)
  ✅ Can fast-forward to trunk
  📊 Change size: 5 files, ~120 lines
  🎉 All checks passed - ready to promote!
```

Now you can promote with confidence:

```
$ sologit pad promote <pad-id>
```

Best Practices

1. Run Fast Tests First

```
# Quick feedback loop
$ sologit pad auto-merge <pad-id> --target fast

# After merge, full tests run via CI
```

2. Use Meaningful Workpad Titles

```
# Good
$ sologit pad create --title "fix-login-timeout"
$ sologit pad create --title "add-payment-webhook"

# Less useful
$ sologit pad create --title "changes"
$ sologit pad create --title "wip"
```

3. Keep Changes Small

Smaller changes = faster reviews = higher confidence

```
# Check change size
$ sologit pad diff <pad-id> --stat
```

4. Monitor CI Results

```
# Watch CI status
$ sologit ci status my-app --follow
```

5. Clean Up Old Workpads

```
# List all workpads
$ sologit pad list my-app

# Delete old ones
$ sologit pad delete <pad-id>
```

Troubleshooting

Problem: Tests Pass Locally But Fail in Auto-Merge

Cause: Environment differences (dependencies, configs, etc.)

Solution:

```
# Check test environment
$ sologit test run <pad-id> --verbose

# Or run tests manually first
$ cd $(sologit pad path <pad-id>)
$ pytest tests/
```

Problem: Promotion Gate Rejects Even Though Tests Pass

Cause: Other gate rules violated (change size, fast-forward, etc.)

Solution:

```
# Check gate decision
$ sologit pad evaluate <pad-id>

# Review reasons
# Adjust rules if needed
```

Problem: CI Smoke Tests Keep Failing

Cause: Integration or environment issues

Solution:

```
# Run smoke tests locally
$ sologit ci smoke my-app --local

# Check logs
$ sologit ci logs my-app

# Fix issues and retest
```

Summary

Phase 3 provides a complete auto-merge workflow:

Feature	Command	Purpose
Auto-merge	`pad auto-merge`	Test + promote in one step
Evaluate	`pad evaluate`	Check promotion readiness
CI smoke	`ci smoke`	Post-merge validation
Rollback	`ci rollback`	Revert failed commits
Configure	Edit config.yaml	Set promotion rules

Philosophy: Tests are the review. If tests pass, code ships.

Next Steps

1. **Try it:** Run through the Quick Start
2. **Configure:** Set up your promotion rules
3. **Integrate:** Configure CI smoke tests
4. **Explore:** Check out the Phase 3 demo (../../examples/phase3_demo.py)

Last Updated: October 17, 2025
Phase: 3 - Auto-Merge & CI Orchestration
Status: Complete ✅