Git Workflow Setup Guide

This guide explains how to set up git workflow tools for Contest Log Analytics, including pre-commit hooks and git aliases for clean history viewing.

Overview

The git workflow setup includes:

- **Pre-commit hooks** that automatically update git hash in `version.py`
- **Git aliases** for viewing clean, linear-looking history
- **Version management** via `contest_tools/version.py`

Key Point: All setup scripts can be run from **Command Prompt (cmd.exe)** - they automatically invoke PowerShell if needed. You don't need to manually open PowerShell.

Quick Start

First Time Setup

Option 1: Complete Setup (Recommended)

```
cd Contest-Log-Analyzer
setup.bat --setup-git
```

This sets up:

- [OK] Development environment (env vars, conda)
- [OK] Pre-commit hooks
- [OK] Git aliases
- [OK] Version management

Option 2: Just Git Workflow Tools

```
cd Contest-Log-Analyzer
tools\setup_git_workflow.bat
```

Detailed Tutorial

Tutorial 1: New Developer / New Machine

Scenario: You just cloned the repository and want to set everything up.

Steps:

1. **Open Command Prompt** (not PowerShell - though that works too)

```
# Navigate to repository
cd c:\Users\YourUser\Repos\Contest-Log-Analyzer
```

2. **Run the setup script**

```
setup.bat --setup-git
```

3. **Verify it worked**

```
# Test git aliases
git mainlog

# Should show clean commit history
# If you see an error, the aliases weren't set up
```

4. **Install pre-commit (if not already installed)**

```
pip install pre-commit
# Then run setup again
setup.bat --setup-git
```

Expected Output:

```
Setting up Git Workflow Tools for Contest Log Analytics...

Step 1: Installing pre-commit hooks...
  [OK] pre-commit is installed
  [OK] Pre-commit hooks installed successfully

Step 2: Setting up git aliases...
  [OK] mainlog alias configured
  [OK] fullhistory alias configured
  [OK] featurelog alias configured
  [OK] recent alias configured
  [OK] branchdiff alias configured
  [OK] Git aliases configured successfully

Step 3: Verifying version.py...
  [OK] version.py exists
  [OK] version.py is functional: 0.160.0-beta.1 (commit abc1234)

Setup Summary:
  * Pre-commit hooks: Configured
  * Git aliases: Configured
  * Version management: Ready
```

Tutorial 2: Daily Use - Viewing Clean History

Scenario: You want to see the main branch history without merge bubbles.

Before Setup:

```
git log --oneline
# Shows all commits including merge commits - can be messy
```

After Setup:

```
git mainlog
# Shows clean, linear history - only merge commits (your squash-style view)
```

Available Commands:

```
git mainlog      # Clean main branch view (recommended)
git recent       # Last 20 commits on main
git fullhistory  # Everything with all branches
git featurelog   # All branches with structure
git branchdiff   # What's in your branch vs main
```

Tutorial 3: Pre-commit Hooks in Action

Scenario: You make a commit and want to see the git hash automatically update.

Steps:

1. **Make a code change**

   ```
   # Edit a file
   notepad contest_tools\version.py
   ```

2. **Stage and commit**

   ```
   git add contest_tools\version.py
   git commit -m "test: verify pre-commit hook"
   ```

3. **Check what happened**

   ```
   # The pre-commit hook automatically:
   # - Updated __git_hash__ in version.py
   # - Added version.py to the commit

   # Verify:
   git show HEAD:contest_tools/version.py | findstr git_hash
   # Should show the current commit hash
   ```

If hooks aren't working:

```
# Reinstall hooks
pre-commit install

# Or re-run setup
tools\setup_git_workflow.bat
```

Tutorial 4: Working with Feature Branches

Scenario: You're working on a feature branch and want to see what's different from main.

Steps:

1. **Create and work on a feature branch**

   ```
   git checkout -b feat/new-widget
   # ... make changes ...
   git commit -m "wip: initial widget code"
   git commit -m "wip: add styling"
   ```

2. **See what's in your branch vs main**

   ```
   git branchdiff
   # Shows commits in feat/new-widget that aren't in main
   ```

3. **When ready, merge to main**

   ```
   git checkout main
   git merge --no-ff feat/new-widget
   # Editor opens: write merge commit message
   ```

4. **View clean history after merge**

   ```
   git mainlog
   # Shows one merge commit for your feature
   # Clean, linear view
   ```

How It Works

The Script Chain

When you run `setup.bat --setup-git` or `tools\setup_git_workflow.bat` from Command Prompt:

```
Command Prompt (cmd.exe)
  ↓
setup.bat or setup_git_workflow.bat
  ↓
```

```
Detects PowerShell availability
  ↓
If PowerShell found:
  -> Invokes: powershell -File tools\setup_git_workflow.ps1
    -> PowerShell does all the work:
      - Installs pre-commit hooks
      - Configures git aliases
      - Verifies version.py
If PowerShell not found:
  -> Uses basic batch commands (limited functionality)
```

You don't need to:

- Manually open PowerShell
- Change execution policy
- Run scripts separately

You just:

- Run the batch file from Command Prompt
- Everything happens automatically

Common Use Cases

Use Case 1: "I just want clean git history"

```
# Run once:
tools\setup_git_workflow.bat

# Now use:
git mainlog    # See clean history anytime
```

Use Case 2: "My pre-commit hook stopped working"

```
# Reinstall hooks:
pre-commit install

# Or full reset:
tools\setup_git_workflow.bat
```

Use Case 3: "I'm on a new computer"

```
# Clone repo, then:
setup.bat --setup-git

# One command does everything
```

Use Case 4: "I want to see version info"

```
python -c "from contest_tools.version import get_version_string; print(get_version_string())
# Output: 0.160.0-beta.1 (commit abc1234)
```

Viewing All Files in Git Status

By default, `git status` groups untracked files in directories together, showing `tools/` instead of listing each file individually. To see all files individually:

```
git status --untracked-files=all
```

This is useful when:

- You want to see exactly which files are untracked
- You're reviewing changes and want the full list
- You're verifying that all expected files are present

Example:

```
# Default view (groups by directory):
git status
# Shows: ?? tools/

# Detailed view (shows all files):
git status --untracked-files=all
# Shows:
# ?? tools/README.md
# ?? tools/setup_git_workflow.bat
# ?? tools/setup_git_workflow.ps1
# ... (all files listed individually)
```

Troubleshooting

Problem: "PowerShell not found" Warning

What it means: The script can't find PowerShell, so it's using basic batch commands.

Solution: This is usually fine! The git aliases will still work. For full functionality:

- PowerShell 5.1+ comes with Windows 10/11
- If missing, install Windows Management Framework 5.1

Problem: Git aliases don't work

Check:

```
git config --list | findstr alias
```

Fix:

```
# Re-run setup
tools\setup_git_workflow.bat
```

Problem: Pre-commit hook not running

Check:

```
pre-commit --version
# If not found: pip install pre-commit
```

Fix:

```
pre-commit install
# Or:
tools\setup_git_workflow.bat
```

Problem: Script says "version.py not found"

Check:

```
dir contest_tools\version.py
```

Fix:

- Make sure you're in the repository root
- The file should exist - if not, check if it was deleted

Problem: "Access Denied" errors

Cause: Permission issues

Fix:

- Run Command Prompt "As Administrator"
- Check file/folder permissions
- On corporate computers, you may need IT support

Advanced: Manual Configuration

If you prefer manual setup or the scripts don't work:

Manual Pre-commit Hook Setup

```
pip install pre-commit
pre-commit install
```

Manual Git Alias Setup

```
git config alias.mainlog "log --first-parent --oneline --graph --decorate"
git config alias.recent "log --first-parent --oneline --graph -20"
git config alias.fullhistory "log --all --graph --oneline --decorate"
git config alias.featurelog "log --oneline --graph --all --decorate"
git config alias.branchdiff "log --oneline --graph main.."
```

Verify Manual Setup

```
# Check aliases
git mainlog

# Check hooks
pre-commit run --all-files

# Check version
python -c "from contest_tools.version import get_version_string; print(get_version_string())
```

Related Documentation

- **`Docs/Contributing.md`** - Full git workflow and commit standards
- **`Docs/AI_AGENT_RULES.md`** - Rules for AI agents handling git operations
- **`tools/README.md`** - Quick reference for setup scripts

Summary

To set up git workflow tools:

1. Open Command Prompt (cmd.exe)
2. Navigate to repository root
3. Run: `setup.bat --setup-git` or `tools\setup_git_workflow.bat`
4. Done! Scripts automatically handle PowerShell invocation.

To use the tools:

- `git mainlog` - View clean history (daily use)
- Pre-commit hooks auto-update git hash before commits
- Version info available via `contest_tools.version.get_version_string()`

No PowerShell needed - just run the batch files from Command Prompt!