Resolving GitHub Desktop commit-blocking errors on Windows

The "Function not implemented" error combined with CRLF warnings that prevent commits represents a convergence of Windows file system limitations, Git's Unix heritage, and GitHub Desktop's architecture. When these issues affect multiple applications, they typically stem from system-level configuration problems that cascade across development tools.

The root cause behind "Function not implemented" errors

The `warning: could not open directory 'env/lib64/': Function not implemented` error occurs when Git for Windows encounters symbolic links or junction points it cannot properly process. (GitHub +4) In Python virtual environments, **lib64 is typically a symbolic link to the lib directory** for Linux compatibility, but Windows Git's readlink() function implementation cannot handle these POSIX-style symlinks correctly. (Stack Overflow +2) This becomes particularly problematic with Git versions 2.16.1-2.16.2 and 2.20.1.windows.1, which have documented regressions in symbolic link handling. (Stack Overflow +2)

The error intensifies when repositories are located on **network drives or cloud-synced folders**, where Windows file system semantics differ from local NTFS drives. (GitHub +3) Additionally, Git's file system cache (fscache) introduced for performance optimization conflicts with these non-standard directory structures, causing the "Function not implemented" message. (GitHub) (Stack Overflow)

Why CRLF issues completely block commits instead of just warning

The critical configuration setting **`core.safecrlf=true`** transforms line ending warnings into blocking errors. When Git detects irreversible line ending conversions—particularly in mixed-format repositories where some files have CRLF and others have LF—it refuses to proceed with commits to prevent data corruption. (Adaptive Patchwork) This becomes a system-wide issue when global Git configuration uses `core.autocrlf=true` on Windows, attempting to convert all text files, which then conflicts with tools expecting specific line endings like shell scripts requiring LF or batch files requiring CRLF. (Edwardthomson)

GitHub Desktop compounds this problem by inheriting system Git settings while adding its own authentication trampolines and submodule handling, creating multiple failure points. (GitHub +4) Versions **2.7.2, 3.3.16-beta1, and 3.3.18** have documented regressions where these systems conflict. (GitHub)

Immediate fixes that resolve both issues

Disable problematic Git features temporarily

The fastest resolution involves disabling the specific Git features causing conflicts. Open Git Bash or Command Prompt and run:

```bash
```

```
git config --global core.fscache false
git config --global core.safecrlf warn
git config --global core.symlinks false
```

These commands disable the file system cache that struggles with symbolic links, convert blocking CRLF errors back to warnings, and stop Git from attempting to process symbolic links. Adaptive Patchwork Stack Overflow **This immediately resolves the "Function not implemented" error** for most users. GitHub Stack Overflow

Fix Python virtual environment structures

For the env/lib64 directory issue specifically, recreate your virtual environment without symbolic links:

```bash
# Remove the problematic virtual environment
rmdir /s env

# Create new environment with copies instead of symlinks
python -m venv --copies env
```

The `--copies` flag forces Python to create actual file copies rather than symbolic links, completely avoiding the Windows Git limitation. Python documentation

Move repositories to local drives

If your repository is on a network drive, OneDrive, or other cloud-synced location, **moving it to a local C:\ drive resolves many issues immediately**. Stack Overflow Network drives lack full POSIX semantics support, and cloud sync services can interfere with Git operations: GitHub +3

```bash
# Move repository to local drive
robocopy "\\network\path\repo" "C:\projects\repo" /E /MOVE
cd C:\projects\repo
git status
```

System-wide Windows configuration fixes

Enable long path support system-wide

Windows' 260-character path limit contributes to many Git errors. github Enable long path support through both Windows and Git: Atlassian Support atlassian

```powershell
```

```
# PowerShell as Administrator
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem" `
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force

# Configure Git for long paths
git config --global core.longpaths true
```

Configure Windows Defender exclusions

Windows Defender scanning can interfere with Git operations, particularly with symbolic links and junction points. Add development directories to exclusions: github +2

```powershell
# PowerShell as Administrator
$devPaths = @("C:\projects", "$env:USERPROFILE\.git", "$env:LOCALAPPDATA\GitHubDesktop")
foreach ($path in $devPaths) {
  Add-MpPreference -ExclusionPath $path
}
Add-MpPreference -ExclusionProcess "git.exe"
Add-MpPreference -ExclusionProcess "GitHubDesktop.exe"
```

Enable developer mode for symbolic links

Windows Developer Mode allows creation of symbolic links without administrator privileges: SciVision +2

```powershell
# PowerShell as Administrator
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock" /t REG_DWORD /
```

Comprehensive CRLF line ending solution

Create a .gitattributes file for consistent handling

Instead of relying on individual developer configurations, **create a .gitattributes file** in your repository root that enforces consistent line endings: Edwardthomson Aleksandr Hovhannisyan

```gitattributes
```

```
# Normalize all text files to LF in repository
* text=auto

# Windows-specific files must have CRLF
*.bat text eol=crlf
*.cmd text eol=crlf
*.ps1 text eol=crlf

# Unix scripts must have LF
*.sh text eol=lf
*.bash text eol=lf

# Binary files should never be modified
*.exe binary
*.dll binary
*.pyc binary
```

Normalize existing repository files

After creating .gitattributes, normalize all existing files to fix mixed line endings: Stack Overflow GitHub

```bash
# Add .gitattributes first
git add .gitattributes
git commit -m "Add line ending configuration"

# Normalize all files (Git 2.16+)
git add --renormalize .
git commit -m "Normalize all line endings"
```

GitHub Desktop specific solutions

Reset GitHub Desktop completely

When GitHub Desktop itself is corrupted or misconfigured:

```powershell
```

```
# Close GitHub Desktop
taskkill /f /im "GitHub Desktop.exe"

# Clear all configuration
Remove-Item -Recurse -Force "$env:APPDATA\GitHub Desktop"
Remove-Item -Recurse -Force "$env:LOCALAPPDATA\GitHubDesktop"

# Restart GitHub Desktop and re-authenticate
```

Version-specific workarounds

Problematic versions to avoid: 2.7.2, 3.3.16-beta1, and 3.3.18 have documented issues with "Function not implemented" errors. GitHub **Recommended stable versions** include 3.4.19 or 3.3.14. Stack Overflow To downgrade:

1. Uninstall current GitHub Desktop
2. Download older version from GitHub releases archive
3. Install and disable auto-updates in preferences

Alternative Git clients when GitHub Desktop fails

If problems persist, consider switching to:

- **Fork**: Fast, native performance with excellent merge tools Git Tower
- **GitKraken**: Professional features with good cross-platform support Git Tower
- **TortoiseGit**: Deep Windows integration through context menus GitKraken
- **Command line with VS Code**: Most reliable, with visual diff/merge in VS Code

Prevention strategies for teams

Standardize development environment

Create a team-wide Git configuration standard:

```bash
# Team .gitconfig template
git config --global core.autocrlf input
git config --global core.safecrlf warn
git config --global core.fscache false
git config --global core.longpaths true
```

Use Docker or WSL2 for complex projects

For projects with extensive symbolic link usage or cross-platform requirements, **Windows Subsystem for Linux 2 (WSL2)** provides a full Linux environment that handles symbolic links correctly: CyberPanel +2

```bash
# Install WSL2
wsl --install

# Clone and work within WSL2
wsl
cd /home/username
git clone https://github.com/your/repo.git
```

Verification and monitoring

After applying fixes, verify resolution:

```bash
# Test Git operations
git status
git add .
git commit -m "Test commit after fixes"

# Check applied configurations
git config --list | grep -E "(fscache|safecrlf|symlinks|longpaths|autocrlf)"

# Verify no symbolic link errors
git ls-files | xargs -I {} git check-attr -a {}
```

These solutions address both the immediate symptoms and underlying causes of the "Function not implemented" and CRLF commit-blocking errors. The combination of disabling problematic Git features, properly configuring line endings through .gitattributes, and ensuring repositories are on local drives with proper Windows settings resolves the vast majority of cases. Stack Overflow For persistent issues, switching to WSL2 or alternative Git clients provides reliable workarounds while maintaining productivity.