🔄 Auto-Commit Setup for GitHub Copilot Training

Overview

Setting up automatic commits helps track every change made during GitHub Copilot exercises, making it easy to revert to previous states and understand the progression of development.

🚀 Recommended Approaches

1. VS Code Extension: GitDoc (Recommended)

Best for: Real-time automatic commits on every save

```
vsls-contrib.gitdoc
```

Setup:

1. Install GitDoc extension
2. Open Command Palette (`Cmd+Shift+P`)
3. Run `GitDoc: Enable`
4. Configure auto-commit settings

Benefits:

- Commits automatically on save
- Works seamlessly with GitHub Copilot
- Maintains detailed commit history
- Easy to enable/disable per project

2. VS Code Extension: Git Automator

Best for: Automated commit messages and workflow

```
ivangabriele.vscode-git-add-and-commit
```

Features:

- Custom commit message templates
- Keyboard shortcuts for quick commits
- Workflow automation

3. AI-Powered Commit Messages

Best for: Intelligent commit message generation

```
michaelcurrin.auto-commit-msg,sitoi.ai-commit
```

Features:

- Auto-generates meaningful commit messages
- Analyzes file changes for context
- Conventional commit format support

🛠️ Manual Git Hooks Setup

Pre-Commit Hook for Training Sessions

Create a custom pre-commit hook that automatically stages and commits Copilot-related changes:

```
# Create pre-commit hook
cat > .git/hooks/pre-commit << 'EOF'
#!/bin/bash

# Auto-commit training changes
if [ -f ".copilot-training-mode" ]; then
    echo "🤖 Copilot Training Mode: Auto-committing changes"

    # Add all changes
    git add .

    # Create commit with timestamp
    TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")
    git commit -m "🤖 Copilot Training Progress - $TIMESTAMP" --no-verify
fi
EOF
```

```
# Make executable
chmod +x .git/hooks/pre-commit
```

To enable training mode:

```
touch .copilot-training-mode
```

To disable training mode:

```
rm .copilot-training-mode
```

📝 VS Code Settings for Auto-Save + Auto-Commit

Add these settings to your workspace or user settings:

```
{
    // Auto-save files for faster commits
    "files.autoSave": "onFocusChange",
    "files.autoSaveDelay": 1000,

    // Git settings
    "git.autofetch": true,
    "git.autoStash": true,
    "git.enableCommitSigning": false,

    // GitDoc specific settings (if using GitDoc extension)
    "gitdoc.enabled": true,
    "gitdoc.commitValidationLevel": "none",
    "gitdoc.commitMessageFormat": "🤖 Training: {filename} - {now}",
    "gitdoc.pullOnOpen": false,
    "gitdoc.pushOnSave": false,
    "gitdoc.commitOnSave": true
}
```

🎯 Training-Specific Commit Strategy

Commit Message Conventions

Use consistent prefixes to track different types of changes:

- `🤖 Copilot:` - Changes made with Copilot assistance

- `✏️ Manual:` - Manual code changes
- `🧪 Exercise:` - Specific exercise completions
- `🔧 Setup:` - Configuration and setup changes
- `🐛 Fix:` - Bug fixes during exercises
- `📚 Docs:` - Documentation updates

Example Workflow Script

Create a training helper script:

```
#!/bin/bash
# save as: copilot-commit.sh

EXERCISE_NAME="$1"
DESCRIPTION="$2"

if [ -z "$EXERCISE_NAME" ]; then
    echo "Usage: ./copilot-commit.sh <exercise-name> [description]"
    exit 1
fi

# Add all changes
git add .

# Create detailed commit message
TIMESTAMP=$(date "+%H:%M:%S")
if [ -n "$DESCRIPTION" ]; then
    COMMIT_MSG="🤖 Exercise: $EXERCISE_NAME - $DESCRIPTION [$TIMESTAMP]"
else
    COMMIT_MSG="🤖 Exercise: $EXERCISE_NAME completed [$TIMESTAMP]"
fi

# Commit with message
git commit -m "$COMMIT_MSG"

echo "✅ Committed: $COMMIT_MSG"
```

Usage:

```
./copilot-commit.sh "interface-basics" "Completed keyboard shortcuts
practice"
./copilot-commit.sh "entity-creation" "Added User entity with JPA
annotations"
```

🔍 Commit History Navigation

Useful Git Commands for Training

```
# View training session commits
git log --oneline --grep="🤖"

# View commits from last hour
git log --since="1 hour ago" --oneline

# View commits for specific file
git log --oneline -- src/main/java/com/taskmanager/app/entity/User.java

# Create checkpoint tags
git tag -a "checkpoint-$(date +%Y%m%d-%H%M)" -m "Training checkpoint"

# Quick revert to previous commit
git reset --hard HEAD~1

# View changes in last commit
git show HEAD
```

VS Code Git Timeline

Use VS Code's built-in Git Timeline feature:

1. Open Source Control panel (`Ctrl+Shift+G`)
2. Enable Timeline view in Explorer
3. See file-by-file commit history
4. Right-click commits to revert changes

⚠️ Best Practices

Do's

- ✅ Use descriptive commit messages with emojis
- ✅ Enable auto-commit only during training sessions
- ✅ Create checkpoint tags at major milestones
- ✅ Review commit history regularly

- ✅ Use consistent message formatting

Don'ts

- ❌ Auto-commit sensitive information
- ❌ Leave auto-commit enabled for production code
- ❌ Commit incomplete or broken code frequently
- ❌ Forget to disable auto-commit after training
- ❌ Auto-push to shared repositories without review

🔄 Reverting Changes

Quick Revert Options

```
# Undo last commit (keep changes)
git reset --soft HEAD~1

# Undo last commit (discard changes)
git reset --hard HEAD~1

# Revert specific file to previous version
git checkout HEAD~1 -- path/to/file.java

# Interactive revert (choose commits)
git rebase -i HEAD~10
```

VS Code Revert

1. Open Source Control panel
2. Right-click on commit in timeline
3. Select "Revert Changes in Commit"
4. Or use "Reset to Commit" for harder reset

This setup ensures every Copilot-assisted change is tracked and easily reversible! 🎯

📍 Creating Restore Checkpoints

1. Use Timeline View (Local History) - Best for Automatic Checkpoints

VS Code automatically saves a history of your changes every time you save a file.

Save Your File: Before significant changes, save the file (`Ctrl+S` or `Cmd+S`). This creates a checkpoint in local history.

Open the Timeline: In the Explorer panel, see the "Timeline" view at the bottom. Or use Command Palette: "Timeline: Show Timeline".

Restore a Checkpoint: Click any previous save to see a diff view and use the Restore button to revert.

2. Use Git Commits - Best for Major Checkpoints

For significant checkpoints (e.g., before refactoring with Copilot):

Stage and Commit:

1. Go to Source Control view (branching icon in sidebar)
2. Stage changes with the '+' icon
3. Write commit message: "Checkpoint before Copilot refactoring"
4. Click checkmark to commit

Revert if Needed: Right-click on changed file → "Discard Changes"

3. Use Undo - Best for Immediate Reversal

If you accept a Copilot suggestion and immediately want to revert:

- **Windows/Linux:** `Ctrl+Z`
- **Mac:** `Cmd+Z`

Recommended Checkpoint Workflow

Change Size	Method
Small changes	Undo (`Ctrl+Z`)
Medium changes	Timeline view (Local History)

Change Size	Method
Major changes	Git commits