

Git, GitHub, and VS Code

Agentic AI for Project Management and Research Productivity

Tomas Larroucau

Arizona State University

February 10, 2026

Workshop Overview

- 1 Introduction
 - Project Structure
- 2 VS Code: Your Research Environment
- 3 Git: Version Control Fundamentals
- 4 GitHub: Collaboration Platform
- 5 Agentic AI for Research

Why This Workshop?

The Problem:

- Research projects grow messy fast
- Co-authoring via email and Dropbox can be chaotic
- AI tools exist but live *outside* the workflow

What If We Could:

- Work in one environment: code, writing, collaboration
- Track every change and undo any mistake
- Let AI operate *inside* the project with guardrails

Goal: Set up the infrastructure for truly integrated, AI-assisted research.

“We’ll let the AI roam free... but set up proper guardrails.”

The “Jur-AI-ssic Park” Approach to Research

The Park

(VS Code)



The island where everything lives

The “Jur-AI-ssic Park” Approach to Research

The Park

(VS Code)



The
Environment

The island where everything lives

The Dinosaurs

(AI Agents)



Raw
Power

Powerful, fast, but chaotic

The “Jur-AI-ssic Park” Approach to Research

The Park

(VS Code)

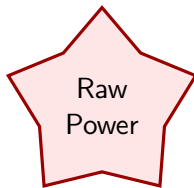


The
Environment

The island where everything lives

The Dinosaurs

(AI Agents)



Raw
Power

Powerful, fast, but chaotic

The Guardrails

(Git & GitHub)



Safety
System

Keeps the chaos contained

The “Jur-AI-ssic Park” Approach to Research

The Park

(VS Code)



The
Environment

The island where everything lives

The Dinosaurs

(AI Agents)



Raw
Power

Powerful, fast, but chaotic

The Guardrails

(Git & GitHub)



Safety
System

Keeps the chaos contained

*“Your scientists were so preoccupied with whether or not they could, they didn’t stop to think if they should... **Git lets you undo if they shouldn’t.**”*

The Symbiosis: Agents as Builders



From Safety to Superpowers:

- **VS Code → Capability**

Agents gain *hands*. They can execute tests, debug, and browse the web.

- **Git/GitHub → Memory**

Issues & PRs provide persistent context ("Pseudo-Memory") and documentation.

- **Agents → Mastery**

They lower the cost of learning the tools.
"How do I cherry-pick this commit?"

What You'll Learn

- **VS Code** as integrated development environment
- **Git fundamentals**: commits, branches, merges
- **GitHub workflows**: issues, pull requests, project boards
- **VS Code Chat**: Ask, Edit, and Agent modes
- **GitHub Copilot**: local and cloud workflows
- **AI-assisted** code review and refactoring
- **Complementary tools**: Refine, NotebookLM, Notion

Workshop Materials

- **Template Repository:** Complete research project structure
- **Sample Data:** CSV files for demonstration
- **Python Scripts:** Analysis, plotting, table generation
- **LaTeX Templates:** Paper and slides
- **Makefile:** Automated workflow
- **Documentation:** README, agent instructions

Repository available at: https://github.com/tlarroucau/AI_workshop

Reproducible Research Template

Repository Structure:

- `data/`: Usually only processed data
- `scripts/`: Analysis code
- `output/`: Generated figures and tables
- `tex/`: LaTeX documents (paper, slides)
- `Makefile`: Automation workflow
- `README.md`: Project documentation
- `.gitignore`: Excluded files

Key Principle: Everything generated from scripts, nothing manual!

See Makefile slide (Appendix)

Workshop Prerequisites

1. Accounts

- **GitHub Account:** Create at github.com
- **Education Benefits:** Apply for student/educator pack at education.github.com (Upgrade to Pro!)

2. Software

- **VS Code:** Download from code.visualstudio.com
- **Git:** Install from git-scm.com
- *Optional:* Python & LaTeX distribution

3. Extensions & Setup

- **Link GitHub:** Sign in to GitHub within VS Code (Accounts menu)
- **Copilot:** Install “GitHub Copilot” extension
- **MCP Servers:** We will use Github MCP server for integration (I will show how)

Note on Access

GitHub Copilot is now free! However, upgrading to Copilot Pro (via Education pack) offers better models and limits. Verification can take a few days.

The Environment

Visual Studio Code



What is VS Code?

Visual Studio Code

- Free, open-source editor by Microsoft
- Cross-platform (Windows, Mac, Linux)
- Extensible via marketplace
- Integrated terminal
- Built-in Git support
- AI assistant integration

Why VS Code for Research?

- Write code *and* papers in one place
- Manage entire project lifecycle
- Collaborate seamlessly
- Automate repetitive tasks
- Leverage AI for productivity

Similar AI Editors: VsCode → Cursor → Windsurf → Antigravity

Essential VS Code Features

Navigation & Interface:

- **Ctrl/Cmd+Shift+P**: Command Palette
- **Ctrl/Cmd+P**: Quick file open
- **Ctrl+`**: Toggle terminal

Multi-Cursor Magic:

- **Ctrl/Cmd+D**: Select next occurrence
- **Alt+Click**: Add cursor anywhere
- **Ctrl/Cmd+Shift+L**: Select all occurrences

Code Editing:

- **Alt+↑/↓**: Move line up/down
- **Ctrl/Cmd+/****: Toggle comment
- **Ctrl/Cmd+Shift+K**: Delete line
- **Ctrl/Cmd+Shift+D**: Duplicate line

Code Navigation:

- **Ctrl/Cmd+Click**: Go to definition
- **Alt+←/→**: Navigate back/forward

Useful Extensions for Research & Productivity

Extension	Purpose
<i>AI & Productivity</i>	
GitHub Copilot	AI code assistant
Codex	Cloud AI coding agent
Continue	Local AI assistant
Cline	Autonomous AI agent
VS Code Speech	Speech-to-text input
<i>Data Science</i>	
Python	Language support
Pylance	IntelliSense, type check
Jupyter	Notebook support
R	R language support
Stata Enhanced	Stata syntax
<i>Documentation</i>	
LaTeX Workshop	Compile LaTeX
LTaX	Grammar/spell check
Markdown All in One	Markdown preview

Extension	Purpose
<i>Project Management</i>	
GitHub PR & Issues	Manage PRs/issues
GitLens	Git visualization
Git History	View git log
Project Manager	Organize projects
Todo Tree	Track TODOs
<i>Data & Utilities</i>	
Rainbow CSV	CSV colorization
Excel Viewer	View Excel files
PDF Viewer	Preview PDFs
Code Snap	Code screenshots
Auto Align	Align code formatting
<i>Development Tools</i>	
Remote SSH	Remote development
Docker	Container support
Error Lens	Inline diagnostics
Live Server	Local web server

Recommended Workspace Structure:

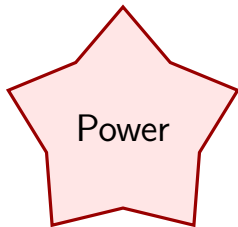
- Open entire project folder as workspace
- Configure settings per workspace (Python path, linters, etc.)
- Save workspace file (.code-workspace) for team sharing

Settings Sync:

- Sync extensions and settings across machines
- Use GitHub or Microsoft account
- Maintain consistency in team environments

Raw Power

AI Agents in Action



What is Agentic AI?

Traditional AI Assistants

- Respond to queries
- Generate code snippets
- Provide suggestions

Agentic AI

- Autonomous task completion
- Multi-step reasoning
- Context-aware assistance

Key Tools:

- GitHub Copilot (VS Code chat modes + Github)
- Cursor, Windsurf, and Antigravity AI Agents
- OpenAI Codex
- Claude Code

VS Code Copilot Chat Modes

1. Ask Mode (Chat Panel)

- Answer questions about code
- Explain complex functions
- Suggest best practices

2. Edit Mode (+ Inline)

- Modify existing code
- Refactor functions
- Apply changes directly

3. Agent Mode

- Multi-file operations
- Workspace-wide changes
- Project scaffolding

4. Plan Mode (New!)

- High-level reasoning
- Break down complex tasks
- Create implementation plans

Note: We can use Local, Background, and Cloud agents in parallel!

VS Code Custom Context & Skills

Custom Instructions

`copilot-instructions.md`

- **Always** in Copilot's context
- Project style & patterns

`AGENTS.md`

- Similar, but only read *on demand*

Prompt Files

`.github/prompts/*.prompt.md`

- Reusable task prompts
- Can specify a model
- Invoked by the user

Agent Skills

`.github/skills/`

- Modular script bundles
- **Progressively loaded** when needed

Demo: `copilot-instructions.md` → `summarize.prompt.md` → `youtube-transcript/skill`

Data Privacy & Security Considerations

Follow Your Institution's Guidelines:

- Always comply with ASU (or your institution's) data security policies
- Review AI tool terms of service for data retention policies!
- Be cautious with sensitive, proprietary, or confidential data

When Cloud AI is Not an Option:

- If data privacy or code confidentiality is a major concern
- If institutional policies prohibit cloud-based AI tools
- If working with regulated data (HIPAA, FERPA, etc.)

Fully Local AI Solutions:

- **Continue + Ollama**: Run local LLMs (Llama, CodeLlama, etc.)
- **Cline** + local models: Autonomous agent without cloud
- Complete offline workflow: No data leaves your machine
- Trade-off: Lower performance than cloud models, but complete privacy

The Guardrails (Local)

Git Version Control



Local
Safety

What is Git?

- **Distributed version control system**
- Tracks changes to files over time
- Enables collaboration without conflicts
- Essential for reproducible research

Why Git for Research?

- Complete history of your work
- Experiment safely with branches
- Collaborate with co-authors
- Publish code alongside papers
- Recover from mistakes

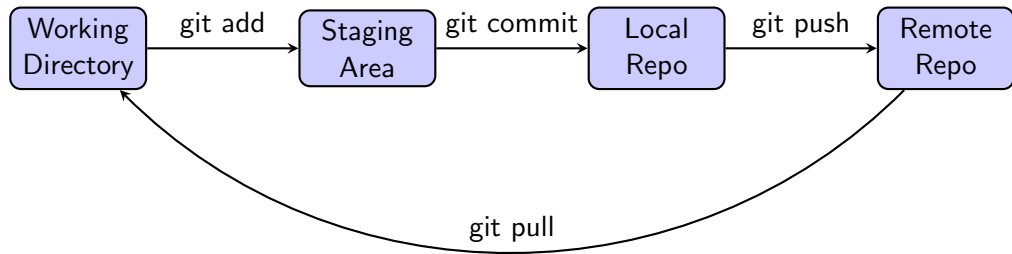
Git Core Concepts

- ① **Repository (repo)**: Project folder tracked by Git
- ② **Commit**: Snapshot of your project at a point in time
- ③ **Branch**: Parallel version of your repository
- ④ **Merge**: Combine changes from different branches
- ⑤ **Remote**: Repository hosted online (e.g., GitHub)

Basic Workflow

```
git add file.py           # Stage changes
git commit -m "message"   # Save snapshot
git push                  # Upload to remote
```

The Git Workflow



- Edit files in working directory
- Stage changes you want to commit
- Commit creates permanent snapshot
- Push shares with collaborators

Essential Git Commands

Repository Setup

<code>git init</code>	# Initialize new repo
<code>git clone <url></code>	# Copy remote repo

Daily Workflow

<code>git status</code>	# Check current state
<code>git add <file></code>	# Stage specific file
<code>git add .</code>	# Stage all changes
<code>git commit -m "msg"</code>	# Commit with message
<code>git fetch</code>	# Download remote changes
<code>git pull</code>	# Fetch + merge changes
<code>git push origin main</code>	# Push to remote
<code>git log</code>	# View commit history
<code>git diff</code>	# See unstaged changes

Branching Strategy

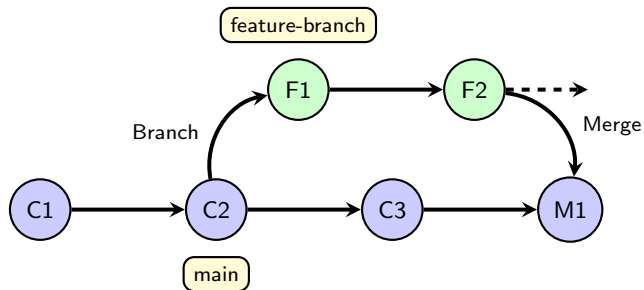
Why branches?

- Isolate experimental work
- Develop features independently
- Keep main branch stable

Common workflow:

- 1 Create branch for new feature: `git checkout -b feature-name`
- 2 Make changes and commit
- 3 Push branch: `git push -u origin feature-name`
- 4 Create pull request on GitHub
- 5 Review and merge
- 6 Delete branch after merge

Visualizing Branches



- **Branch:** Create a parallel timeline (`git checkout -b`)
- **Commit:** Work on the feature branch (F1, F2)
- **Merge:** Combine feature history back into main (`git merge`)
- *Note: The branch remains open after merging unless deleted*

Git Best Practices

Commit Messages

- Be descriptive: "Add regression analysis for Model 2"
- Use imperative mood: "Fix data loading bug"
- Reference issues: "Closes #42"

What to Commit

- **DO**: Source code, scripts, documentation
- **DO**: Raw data (if reasonable size)
- **DON'T**: Generated outputs (rebuild from scripts)
- **DON'T**: Large binary files (use Git LFS if needed)
- **DON'T**: Passwords or API keys

The .gitignore File

What is .gitignore?

- Tells Git which files to ignore (never commit)
- Prevents committing generated files, credentials, or large binaries
- One per repository (in root directory)

Example .gitignore for Research Projects:

```
# Python
__pycache__/
*.pyc
.ipynb_checkpoints/
*.egg-info/

# R
.Rhistory
.RData
.Rproj.user/

# LaTeX
*.aux
*.bbl
*.blg
*.log
*.out
*.fls
*.synctex.gz

# Generated outputs
output/
*.pdf
*.png
```

Common Challenges & Solutions

Understanding Merge Conflicts:

- Occur when Git cannot automatically determine which version to keep
- Happen when the same lines of code/text are edited in different branches
- VS Code highlights conflicts and lets you choose which version to keep
- AI agents can suggest the best resolution strategy!

Backing Up Your Work:

- You can keep your local repo in Dropbox for automatic backup
- Avoid sharing that Dropbox folder with co-authors unless you can really coordinate well

Challenge	Solution
Merge conflicts	Use VS Code conflict resolver, ask AI agent for help
Large files	Use .gitignore, Git LFS if needed
Slow Git operations	Use .gitignore for generated files
Lost work	Commit often, use branches
Unclear AI output	Refine prompts, add context
Reproducibility	Use Makefile, document dependencies

The Guardrails (Cloud)

GitHub Collaboration



Cloud
Memory

What is GitHub?

GitHub is...

- Web-based Git hosting
- Social coding platform
- Project management tools
- Collaboration infrastructure
- Portfolio for researchers

Key Features:

- Remote repository hosting
- Pull requests
- Issues and project boards
- Hosts your personal Web site!
- Pages (documentation)
- Copilot (AI Agents)

Free for academic use! (github.com/education)

Demo: Showing Github and Creating Personal Website

GitHub Workflow: Issues

What are Issues?

- Track tasks, bugs, feature requests
- Organize work with labels and milestones
- Assign to team members
- Reference in commits and PRs

Issue-Driven Development:

- 1 Create issue: “Add robustness checks”
- 2 Create branch: `git checkout -b issue-42-robustness`
- 3 Work on feature, commit with “Addresses #42”
- 4 Create pull request
- 5 Merge and close: “Closes #42”

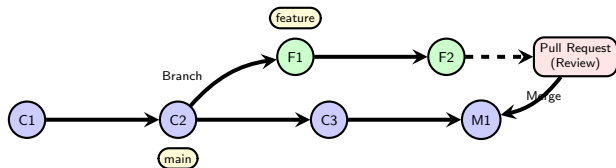
Demo: Creating and assigning an issue to Github Copilot

GitHub Workflow: Pull Requests

What are Pull Requests (PRs)?

- Propose changes to repository
- Enable code review before merging
- Discuss implementation details

PR Workflow:



- 1 Create and push feature branch
- 2 Open Pull Request on GitHub
- 3 Review, discuss, and fix
- 4 Merge when approved

Demo: Creating and reviewing a PR, did Copilot finish working?

GitHub Project Boards

Organize Research Projects:

- Kanban-style boards
- Columns: To Do, In Progress, Done
- Link to issues and PRs
- Automate card movement

Example Workflow:

- 1 Create project board for paper
- 2 Add columns for analysis, writing, revisions
- 3 Create issues for each task
- 4 Move cards as work progresses
- 5 Track progress visually

Demo: Setting up a project board + Creating issue

Integrations

Putting It All Together



MCP: Model Context Protocol

What is MCP?

- Protocol that gives AI agents access to external tools and services
- Enables *direct interaction* with APIs (not just generating code for you)

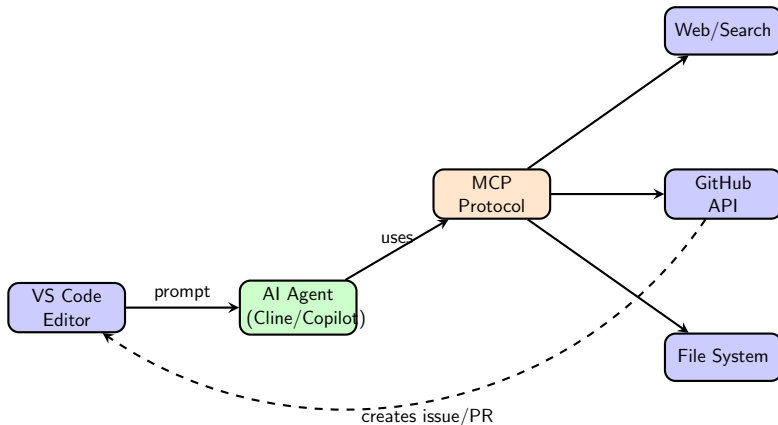
Key MCP Servers:

- **GitHub**: Create issues, PRs, manage boards
- **Playwright**: Browser automation and interaction
- **Filesystem & Fetch**: Built-in VS Code tools

Setup (GitHub MCP):

- 1 Install MCP server (npx)
- 2 Get GitHub token
- 3 Configure in Copilot settings
- 4 Done!

How AI Tools Connect: The Big Picture



Example Flow:

- 1 You: "Create issue for clustered SE analysis and add to project board"
- 2 AI Agent → MCP → GitHub API → Issue created + added to board
- 3 VS Code shows notification with issue link
- 4 Agent can then create branch, write code, commit, create PR automatically!

Live Demo: End-to-End Example

Scenario: Add a new robustness check using MCP-enabled AI agent

- 1 Ask AI agent: "Create issue for adding clustered SE robustness check"
- 2 Agent uses GitHub MCP to create issue and add to project board
- 3 Agent fetches issue details and creates branch: `feature/clustered-se`
- 4 Agent uses Copilot to generate implementation code
- 5 Update analysis script with AI assistance
- 6 Run `make all` to regenerate outputs
- 7 Agent commits changes with descriptive message
- 8 Agent creates PR linking to original issue
- 9 Review PR and merge to main
- 10 Agent updates issue with concise description of changes and closes it

We'll work through this together - from idea to merged code in minutes!

Common Frictions in Academic Research

Integration Challenges:

Statistical Software

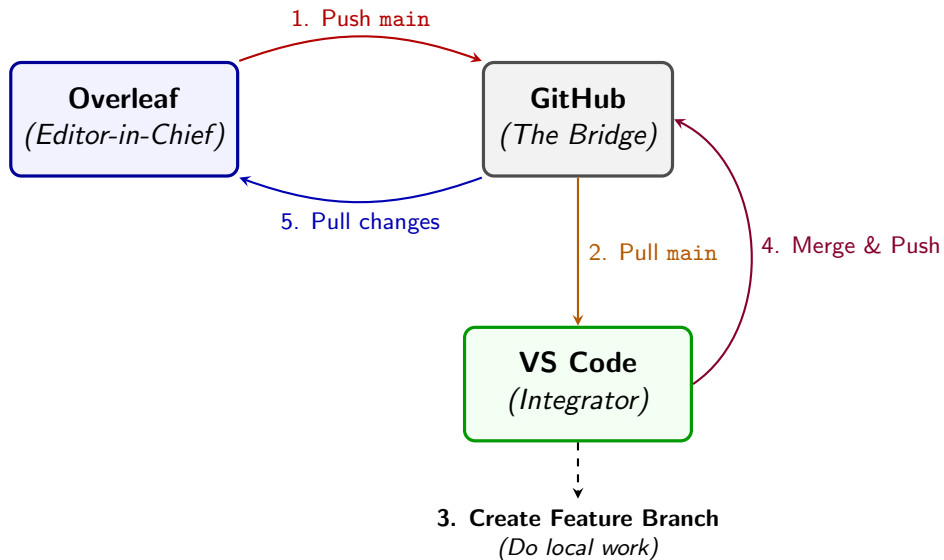
- **Stata/MATLAB:** Not natively integrated in VS Code
- **Solution:** Run through integrated terminal with commands
- Extensions available for syntax highlighting
- Execute code blocks via terminal shortcuts

Overleaf Integration

- **Via Dropbox:** Sync local folder with Overleaf project (some lag)
- **Via Git:** Clone Overleaf project, push/pull changes (more control)
- **Via GitHub:** Sync Overleaf project with GitHub repository (requires Premium)
- **Alternative:** Use LaTeX Workshop extension in VS Code directly and Copy/Paste files

Bottom line: Not perfect, but workable with some adjustments!

Optimum Workflow: Overleaf + VS Code



Complementary Research Tools

Tool	Purpose
Refine (refine.ink)	AI-powered writing revision and style improvement Interactive editing with suggestions
NotebookLM (Google)	Structured reading and note-taking Create study guides from papers Create academic podcasts!
Elicit	Literature discovery and synthesis Extract data from papers
GPT/Claude/Gemini	General research assistance Draft writing, brainstorming

Integration: Use alongside Git/GitHub workflow for complete research pipeline

Next Steps

First:

- 1 Fork/clone workshop template repository
- 2 Install VS Code and extensions
- 3 Set up GitHub account (education benefits)
- 4 Practice basic Git commands
- 5 Run `make all` to build template
- 6 Ask your favorite AI agent to explain any concepts from the slides you don't understand

Second:

- Apply workflow to a small project
- Create repository for current research
- Experiment with Copilot
- Set up project board

Resources

Documentation

- Git: git-scm.com/doc
- GitHub: docs.github.com
- VS Code: code.visualstudio.com/docs

Learning

- GitHub Skills: skills.github.com
- Software Carpentry: software-carpentry.org

Workshop Materials

- Template: github.com/tlarroucau/AI_workshop
- Slides: [tex/slides/](#)

**This ENTIRE workshop was created with ONE prompt and
SOME hours of edits!**

Thank you for attending!

Contact: `Tomas.Larroucau@asu.edu`

Template Repository:

`github.com/tlarroucau/AI_workshop`

The Makefile Approach

What is a Makefile?

A recipe file that automates your workflow. You define *targets*, their *dependencies*, and the *commands* to run.

Syntax

```
target: dependencies
    command to run
```

Then type `make target` in the terminal.

Why it matters for AI: When agents modify your code, `make all` re-runs the full pipeline — ensuring reproducibility even after AI-generated changes.

[Back to Project Structure](#)

Example

```
DATA = data/raw/sample_data.csv
OPTS = --robust --output output/

all: analysis paper

analysis: $(DATA)
    python scripts/analysis.py $(OPTS)

paper:
    cd tex/paper && latexmk paper

clean:
    rm -rf output/*
```

Demo: `make all`