Complete Setup Guide for Claude Desktop + Git + Google Apps Script

For Windows - Partner Edition

Time Required: 45-60 minutes
Skill Level: Beginner-friendly (detailed instructions included)

📋 What You'll Have When Done

- ✅ Claude Desktop with local file editing
- ✅ Automatic Git version control
- ✅ GitHub cloud backup
- ✅ Google Apps Script deployment
- ✅ Everything works with simple commands through Claude

PART 1: Software Installation (20 minutes)

Step 1: Install Node.js

What it is: JavaScript runtime needed for clasp and MCP servers

1. Open your web browser
2. Go to: **https://nodejs.org/**
3. Click the **green "LTS" button** (recommended version)
4. Download will start automatically
5. Once downloaded, **run the installer**
6. **IMPORTANT:** During installation:
   - ✅ Check "Automatically install the necessary tools"
   - ✅ Accept all defaults
   - Click "Next" → "Next" → "Install"
7. Wait for installation (2-3 minutes)
8. **Restart your computer** (this is important!)

Verify it worked:

1. After restart, press `Win + R`
2. Type `cmd` and press Enter
3. In the black window, type: `node --version`
4. You should see something like: `v20.11.0`
5. Also type: `npm --version`
6. You should see something like: `10.2.4`

✅ **If you see version numbers, Node.js is installed correctly!**

Step 2: Install Git

What it is: Version control system to track file changes

1. Go to: **https://git-scm.com/download/win**
2. Download will start automatically (64-bit version)
3. Run the installer
4. **Important settings during installation:**
   - **Select Components:** Keep all defaults checked
   - **Default editor:** Choose "Use Visual Studio Code as Git's default editor"
   - **Adjusting PATH:** Choose "Git from the command line and also from 3rd-party software"
   - **Choosing HTTPS transport:** Choose "Use bundled OpenSSH"
   - **Line ending conversions:** Choose "Checkout Windows-style, commit Unix-style"
   - **Terminal emulator:** Choose "Use Windows' default console window"
   - **Extra options:** Keep defaults
5. Click "Install"
6. Click "Finish"

Verify it worked:

1. Press `Win + R` , type `cmd` and press Enter
2. Type: `git --version`

3. You should see: `git version 2.43.0` (or similar)

✅ **If you see a version number, Git is installed!**

Step 3: Install Claude Desktop

What it is: Desktop app where you'll interact with Claude AI

1. Go to: **https://claude.ai/download**
2. Click **"Download for Windows"**
3. Run the installer: `Claude-Setup-x64.exe`
4. Follow the installation wizard
5. Launch Claude Desktop
6. **Sign in** with your Claude Pro account
   - If you don't have one, create it at claude.ai
   - **You MUST have Claude Pro** ($20/month) for this workflow
7. Close Claude Desktop for now (we'll configure it later)

✅ **Claude Desktop is installed!**

Step 4: Install Visual Studio Code (Optional but Recommended)

What it is: Code editor for manual file editing

1. Go to: **https://code.visualstudio.com/**
2. Click **"Download for Windows"**
3. Run the installer
4. **Important - Check these boxes:**
   - ✅ Add "Open with Code" action to Windows Explorer file context menu
   - ✅ Add "Open with Code" action to Windows Explorer directory context menu
   - ✅ Register Code as an editor for supported file types
   - ✅ Add to PATH
5. Click "Next" → "Install"
6. Launch VS Code when done

✅ **VS Code is installed!**

Step 5: Configure Git (First Time Setup)

What it does: Tells Git who you are for commit messages

1. Press `Win + R`, type `cmd` and press Enter
2. Type these commands (replace with YOUR information):

```
git config --global user.name "Your Full Name"
git config --global user.email "your.email@gmail.com"
```

Example:

```
git config --global user.name "John Smith"
git config --global user.email "john.smith@gmail.com"
```

3. Verify it worked:

```
git config --global user.name
git config --global user.email
```

You should see your name and email printed back.

✅ **Git is configured!**

PART 2: Create GitHub Account & Setup (10 minutes)

Step 6: Create GitHub Account

What it is: Cloud backup for your code

1. Go to: **https://github.com/signup**
2. Enter your email address
3. Create a password
4. Choose a username (this will be public)
5. Verify your email
6. Complete the signup process

✅ **GitHub account created!**

Step 7: Create Personal Access Token (For Git Push)

Why: Git needs permission to push to GitHub

1. Log in to GitHub
2. Click your **profile picture** (top right)
3. Click **"Settings"**
4. Scroll down on the left sidebar
5. Click **"Developer settings"** (near the bottom)
6. Click **"Personal access tokens"**
7. Click **"Tokens (classic)"**
8. Click **"Generate new token"** → **"Generate new token (classic)"**
9. Give it a name: "Git Access Token"
10. Set expiration: **"No expiration"** (or choose a long period)
11. **Check these scopes:**
    - ✅ `repo` (all sub-items)
    - ✅ `workflow`
12. Scroll down and click **"Generate token"**
13. **CRITICAL:** Copy the token that appears (starts with `ghp_`)
14. **Save it in a safe place** (Notepad, password manager)
    - ⚠️ You'll NEVER see this token again!
    - ⚠️ If you lose it, you'll need to create a new one

✅ **GitHub token created and saved!**

Step 8: Create Your First Repository

1. Go to: **https://github.com**
2. Click the **"+"** in the top right
3. Click **"New repository"**
4. Repository name: **`projects`** (lowercase)
5. Description: "My development projects"
6. Choose: **Private** (recommended) or Public
7. ✅ Check **"Add a README file"**
8. Click **"Create repository"**

9. **Copy the repository URL** (should look like):

```
https://github.com/yourusername/projects.git
```

✅ **GitHub repository created!**

PART 3: Setup Google Apps Script Access (5 minutes)

Step 9: Install clasp (Google Apps Script CLI)

What it is: Command-line tool to deploy to Google Apps Script

1. Press `Win + R`, type `cmd` and press Enter
2. Type this command:

```
npm install -g @google/clasp
```

3. Wait for installation (1-2 minutes)
4. Verify it worked:

```
clasp --version
```

5. You should see a version number

✅ **clasp is installed!**

Step 10: Login to clasp

1. In the command prompt, type:

```
clasp login
```

2. Your web browser will open automatically
3. **Sign in with your Google account** (the one you use for Google Apps Script)
4. Click **"Allow"** to give clasp permissions
5. You should see **"Logged in! You may close this page."** in browser
6. In command prompt, you should see: **"Authorization successful."**

✅ **clasp is authenticated with Google!**

PART 4: Setup Your Project Folder (5 minutes)

Step 11: Create Your Projects Folder

1. Open **File Explorer** (Win + E)
2. Navigate to your **D: drive** (or C: if you don't have D:)
3. Right-click in empty space
4. Click **"New"** → **"Folder"**
5. Name it: `Projects`
6. Your path should be: `D:\Projects` (or `C:\Projects`)

Remember this path! You'll use it throughout.

✅ **Projects folder created!**

Step 12: Clone Your GitHub Repository

Method A: Using Command Line (Recommended)

1. Press `Win + R`, type `cmd` and press Enter
2. Navigate to your D: drive:

```
D:
```

3. Clone your repository (replace with YOUR username):

```
git clone https://github.com/yourusername/projects.git Projects
```

4. When prompted for credentials:
   - Username: Your GitHub username
   - Password: **Paste your Personal Access Token** (from Step 7)
5. You should see: "Cloning into 'Projects'..."

✅ **Repository cloned to D:\Projects!**

Step 13: Create Test Project Structure

1. Open File Explorer
2. Navigate to `D:\Projects`
3. Create a new folder called `Test`
4. Your structure should be:

```
D:\Projects\
├── README.md (from GitHub)
└── Test\
```

✅ **Project structure created!**

PART 5: Configure Claude Desktop with MCP (15 minutes)

Step 14: Open Claude Desktop Configuration

1. **Launch Claude Desktop**
2. In the **top menu bar**, click **"Claude"**
3. Click **"Settings..."**
4. In the Settings window, click **"Developer"** tab on the left
5. Click the **"Edit Config"** button

This will open a file in Notepad (or your default text editor).

The file location is:

```
C:\Users\YourUsername\AppData\Roaming\Claude\claude_desktop_config.json
```

✅ **Config file is open!**

Step 15: Add MCP Server Configuration

1. The file is probably empty or has just: `{}`
2. **Delete everything** in the file
3. **Copy and paste this ENTIRE configuration:**

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "D:\\\\Projects"
      ]
    },
    "desktop-commander": {
      "command": "npx",
      "args": [
        "-y",
        "@wonderwhy-er/desktop-commander"
      ]
    }
  }
}
```

⚠️ **CRITICAL NOTES:**

- If your Projects folder is on **C: drive**, change `"D:\\\\Projects"` to `"C:\\\\Projects"`
- Notice the **double backslashes** `\\` - this is required in JSON

- Make sure you copy the ENTIRE thing including the outer `{ }`
- Don't add any extra commas or change any quotes

4. **Save the file** (Ctrl + S)
5. **Close the text editor**

✅ **MCP configuration added!**

Step 16: Restart Claude Desktop

1. **Completely close Claude Desktop**
   - Right-click the Claude icon in system tray (bottom right)
   - Click **"Exit"** or **"Quit"**
2. Wait 5 seconds
3. **Relaunch Claude Desktop**
4. Wait for it to fully load (10-15 seconds)

Step 17: Verify MCP is Working

1. In Claude Desktop, look at the **bottom-right** of the message input box
2. You should see a 🔨 **hammer icon**
3. **Click the hammer icon**
4. You should see a menu with tools like:
   - `read_file`
   - `write_file`
   - `list_directory`
   - `execute_command`
   - `start_process`
   - And many more...

✅ **If you see the hammer icon and these tools, MCP is working!**

❌ **If you DON'T see the hammer icon:**

- Go back to Step 14
- Check the config file for typos
- Make sure paths use `\\` (double backslash)
- Try restarting your computer
- Restart Claude Desktop again

PART 6: Test Everything Works (10 minutes)

Step 18: Test File Access

In Claude Desktop, send this message:

```
Can you list all files and folders in D:\Projects?
```

Claude should:

1. Ask for permission to access the folder
2. Click **"Allow"**
3. Show you what's in the folder (probably just README.md and Test/)

✅ **Filesystem access is working!**

Step 19: Test Terminal Commands

In Claude Desktop, send this message:

```
Check the Git status of D:\Projects
```

Claude should:

1. Ask for permission to run the command
2. Click **"Allow"**
3. Show you the Git status output

✅ **Terminal commands are working!**

Step 20: Create Your First Apps Script Project

In Claude Desktop, send this message:

```
Please do the following:
1. Create a new file D:\Projects\Test\Code.js with a simple function
2. Create D:\Projects\Test\appsscript.json manifest file
3. Initialize a clasp project in the Test folder
4. Show me what you created
```

Claude should:

1. Create the files
2. Ask permissions for each operation
3. Click "Allow" for each
4. Set up the clasp project
5. Show you the files it created

✅ **Google Apps Script integration is working!**

Step 21: Test the Complete Workflow

Now send this message:

```
Update the Code.js file to add a new function that writes to a Google Sheet,
then commit to Git and push to both GitHub and Google Apps Script.
```

Claude should automatically:

1. ✅ Modify the Code.js file
2. ✅ Run `git add .`
3. ✅ Run `git commit -m "Added sheet function"`
4. ✅ Run `git push` (you may need to enter your GitHub credentials)
5. ✅ Run `clasp push`
6. ✅ Tell you it's done!

If Git asks for credentials:

- Username: Your GitHub username
- Password: Your **Personal Access Token** (from Step 7)

✅ **Complete workflow is working!**

PART 7: Daily Usage Guide

How to Use This Every Day

Option 1: Let Claude Do Everything (Easiest)

Just tell Claude in plain English:

```
"Add a function to send emails in Code.js, then commit and deploy"
```

```
"Update the API endpoint in Config.js and push everything"
```

```
"Create a new project folder called ClientX with standard Apps Script files"
```

Claude will:

- Edit the files
- Commit to Git
- Push to GitHub
- Deploy to Google Apps Script
- All automatically!

Option 2: Manual Editing + Claude Deployment

1. Edit files in VS Code
2. Save your changes
3. Tell Claude:

```
"Commit and push my recent changes to Code.js"
```

Claude will handle Git and deployment.

Option 3: Fully Manual (If Claude is Down)

1. Edit files in VS Code
2. Open Command Prompt (`Win + R` → `cmd`)
3. Run these commands:

```
cd D:\Projects
git add .
git commit -m "Description of changes"
git push
cd Test
clasp push
```

Common Tasks

Start a New Client Project

Tell Claude:

```
Create a new folder D:\Projects\ClientName with standard Apps Script files
(Code.js, Config.js, Utilities.js, appsscript.json), initialize Git,
and set up clasp.
```

Deploy Updates

Tell Claude:

```
Push all my changes to Git and deploy the ClientName project
```

Check What Changed

Tell Claude:

```
Show me what files have changed since last commit
```

View Project Status

Tell Claude:

```
Show me the Git status and list all my project folders
```

TROUBLESHOOTING

Problem: Hammer icon doesn't appear in Claude Desktop

Solutions:

1. Check the config file has no typos
2. Make sure paths use `\\` (double backslash)
3. Verify Node.js is installed: `node --version` in cmd
4. Restart Claude Desktop completely
5. Restart your computer
6. Reinstall Claude Desktop

Problem: Git push says "Everything up-to-date" but files changed

Solution: You forgot to **commit**!

```
git add .
git commit -m "Your message"
git push
```

You must do all three steps!

Problem: Git asks for password every time

Solution: Use credential helper:

```
git config --global credential.helper wincred
```

Then push once with your Personal Access Token, and Windows will remember it.

Problem: clasp push fails with "not logged in"

Solution:

```
clasp login
```

Then try again.

Problem: clasp push says "No .clasp.json file"

Solution: You need to initialize the project:

```
cd D:\Projects\YourProject
clasp create --title "Project Name" --type standalone
```

Or tell Claude: "Initialize a clasp project in this folder"

Problem: Git push asks for username/password but token doesn't work

Solution: GitHub changed authentication. Use:

- Username: Your GitHub username
- Password: Your **Personal Access Token** (NOT your GitHub password)

If you lost the token, create a new one (Step 7).

QUICK REFERENCE CARDS

Git Commands

```
git status              # See what changed
git add .               # Stage all changes
git add Test/          # Stage only Test folder
git commit -m "msg"    # Commit with message
git push               # Upload to GitHub
git pull               # Download from GitHub
git log                # See commit history
```

clasp Commands

```
clasp login            # Login to Google
clasp create           # Create new project
clasp push             # Deploy to Google Apps Script
clasp pull             # Download from Google Apps Script
clasp open             # Open project in browser
clasp logs             # View execution logs
```

Common Workflows

Update and deploy a project:

```
cd D:\Projects
git add .
git commit -m "Updated code"
git push
cd Test
clasp push
```

Or just tell Claude:

```
"Commit and deploy Test project"
```

FILE STRUCTURE REFERENCE

```
D:\Projects\                          ← Git repository root
├── .git\                             ← Git database (don't touch)
├── .gitignore                        ← Files to ignore
├── README.md                         ← Project description
│
├── ClientA\                          ← First client
│   ├── .clasp.json                  ← Google Apps Script config
│   ├── appsscript.json              ← Manifest
│   ├── Code.js                      ← Main code
│   ├── Config.js                    ← Configuration
│   └── Utilities.js                 ← Helper functions
│
├── ClientB\                          ← Second client
│   ├── .clasp.json
│   ├── appsscript.json
│   └── Code.js
│
└── Test\                             ← Test/sandbox project
    ├── .clasp.json
    ├── appsscript.json
    └── Code.js
```

CONFIGURATION FILES BACKUP

claude_desktop_config.json

Keep a copy of this for backup:

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "D:\\\\Projects"
      ]
    },
    "desktop-commander": {
      "command": "npx",
      "args": [
        "-y",
        "@wonderwhy-er/desktop-commander"
      ]
    }
  }
}
```

Location:

```
C:\Users\YourUsername\AppData\Roaming\Claude\claude_desktop_config.json
```

.gitignore Template

Create this in `D:\Projects\.gitignore`:

```
# Node modules
node_modules/

# Logs
*.log
logs/

# Environment variables
.env
.env.local

# IDE files
.vscode/
.idea/
```

```
# OS files
.DS_Store
Thumbs.db
desktop.ini

# Build files
dist/
build/
```

FINAL CHECKLIST

Before considering setup complete, verify:

- ✅ Node.js installed and working (`node --version`)
- ✅ Git installed and configured (`git --version`)
- ✅ Claude Desktop installed with Pro subscription
- ✅ VS Code installed (optional but recommended)
- ✅ GitHub account created
- ✅ Personal Access Token saved securely
- ✅ GitHub repository created
- ✅ clasp installed and authenticated
- ✅ D:\Projects folder created
- ✅ Repository cloned to D:\Projects
- ✅ Claude Desktop MCP configured
- ✅ Hammer icon visible in Claude Desktop
- ✅ Filesystem access tested
- ✅ Terminal commands tested
- ✅ Complete workflow tested end-to-end

SUPPORT & HELP

If something doesn't work:

1. **Check this guide** - read the troubleshooting section
2. **Ask Claude** - "I'm getting error X, how do I fix it?"
3. **Check official docs:**
   - Git: https://git-scm.com/doc
   - GitHub: https://docs.github.com/
   - clasp: https://github.com/google/clasp
   - Claude Desktop: https://support.claude.ai/
4. **Contact your partner** - they've been through this!

WHAT'S NEXT?

Now that everything is set up:

1. **Practice the workflow** - Create a few test projects
2. **Organize your projects** - Create folders for each client
3. **Customize your setup** - Add more tools as needed
4. **Learn Git basics** - Understanding commits, branches, etc.
5. **Explore Claude's capabilities** - Try different prompts

🎉 **Congratulations! You're all set up!**

You now have a professional development workflow where Claude can:

- Edit your local files
- Track changes with Git
- Backup to GitHub
- Deploy to Google Apps Script
- All through simple conversation!

Happy coding! 🚀