

2025-11-01

section 0

[2em] Part Overview · Duration:

Beginner-Friendly Visual Study Guide

subsection 0.0 What You'll Learn

- **MCP Architecture:** 12 specialized servers (filesystem, github, pytest, pandas, etc.)
- **Auto-trigger Strategy:** Keyword-based orchestration (NO user confirmation needed)
- **Server Chaining:** Combine 3-5 MCPs for complete workflows
- **Integration:** .mcp.json configuration and debugging workflows

subsection 0.0 Why This Matters

Problem: Manual tool selection slows AI workflows (e.g., "Should I use filesystem or grep?").

Solution: MCPs auto-trigger based on task keywords (data analysis -> pandas-mcp, testing -> pytest-mcp, UI -> puppeteer).

Impact: Phase 5 research (11 tasks) leveraged 8/12 MCPs automatically, reducing manual tool selection by 70%.

section 0 The 12 MCP Servers

subsection 0.0 Core Servers (4)

- **1. filesystem:** File operations (read, write, list, search)
- **2. github:** Issues, PRs, checks, releases
- **3. sequential-thinking:** Planning, debugging, reasoning
- **4. puppeteer:** UI testing, browser automation

subsection 0.0 Development Servers (4)

- **1. pytest-mcp:** Test debugging, coverage analysis
- **2. git-mcp:** Advanced Git operations (rebase, cherry-pick, blame)
- **3. mcp-analyzer:** Code quality, linting, static analysis
- **4. mcp-debugger:** API testing, HTTP requests

subsection 0.0 Data Science Servers (3)

- **1. sqlite-mcp:** PSO optimization database queries
- **2. pandas-mcp:** Data analysis, CSV/JSON processing
- **3. numpy-mcp:** Numerical compute, matrix operations

subsection 0.0 Audit Server (1)

- **1. lighthouse-mcp:** Accessibility audits (WCAG compliance)

section 0 Auto-trigger Keywords

subsection 0.0 Filesystem MCP

Triggers: "read file", "list directory", "search files", "find *.py"

```
lstnumber# User: "Read all controller files"
lstnumber# AI auto-triggers: filesystem.list("src/controllers/")
lstnumber
lstnumber# User: "Find all test files"
lstnumber# AI auto-triggers: filesystem.search("test_*.py")
```

subsection 0.0 GitHub MCP

Triggers: "create PR", "list issues", "check CI status", "release notes"

```

lstnumber# User: "Create PR for LT-7 paper"
lstnumber# AI auto-triggers: github.create_pr(title="feat: Add LT-7 research paper", ...)
lstnumber
lstnumber# User: "List open issues"
lstnumber# AI auto-triggers: github.list_issues(state="open")

```

subsection 0.0 Pytest MCP

Triggers: "run tests", "debug test failure", "coverage report"

```

lstnumber# User: "Why is test_classical_smc failing?"
lstnumber# AI auto-triggers:
    pytest_mcp.debug("tests/test_controllers/test_classical_smc.py")
lstnumber
lstnumber# User: "Show coverage for controllers"
lstnumber# AI auto-triggers: pytest_mcp.coverage("src/controllers/")

```

subsection 0.0 Pandas MCP

Triggers: "analyze CSV", "plot benchmark data", "merge datasets"

```

lstnumber# User: "Analyze PSO convergence data"
lstnumber# AI auto-triggers: pandas_mcp.read_csv("optimization_results/pso_results.csv")
lstnumber# AI auto-triggers: pandas_mcp.plot(df["cost"], title="PSO Convergence")
lstnumber
lstnumber# User: "Merge controller benchmarks"
lstnumber# AI auto-triggers: pandas_mcp.merge(df1, df2, on="controller_name")

```

section 0 Server Chaining Patterns

subsection 0.0 Pattern 1: Data Analysis Pipeline

Goal: Analyze PSO optimization results (CSV -> DataFrame -> Plot)

```

lstnumber# Chain: filesystem -> sqlite -> pandas
lstnumber1. filesystem.read("optimization_results/pso_results.csv")
lstnumber2. sqlite_mcp.query("SELECT * FROM optimizations WHERE cost < 0.1")
lstnumber3. pandas_mcp.analyze(df)
lstnumber4. pandas_mcp.plot(df["iteration"], df["cost"])

```

subsection 0.0 Pattern 2: Testing Workflow

Goal: Debug failing test, analyze coverage, fix issue

```

lstnumber# Chain: pytest-mcp -> puppeteer -> mcp-analyzer
lstnumber1. pytest_mcp.run("tests/test_controllers/")
lstnumber2. pytest_mcp.debug("test_classical_smc::test_convergence")
lstnumber3. puppeteer.screenshot("UI elements for manual inspection")
lstnumber4. mcp_analyzer.lint("src/controllers/classical_smc.py")

```

subsection 0.0 Pattern 3: Research Task Workflow

Goal: Implement LT-7 paper generation (planning -> code -> data -> doc)

```

lstnumber# Chain: sequential-thinking -> filesystem -> pandas -> github
lstnumber1. sequential_thinking.plan("LT-7 research paper structure")
lstnumber2. filesystem.create("academic/paper/publications/lt7_paper.tex")
lstnumber3. pandas_mcp.analyze("benchmarks/processed/comparative_study.csv")
lstnumber4. pandas_mcp.generate_figures(df, output="academic/paper/figures/")
lstnumber5. github.create_pr(title="feat: Add LT-7 research paper")

```

section 0 MCP Configuration

subsection 0.0 Configuration File Structure

File: .mcp.json

```

lstdisplay{
lstdisplay  "mcpServers": {
lstdisplay    "filesystem": {
lstdisplay      "command": "npx",
lstdisplay      "args": ["-y", "@modelcontextprotocol/server-filesystem",
lstdisplay        "D:/Projects/main"]
lstdisplay    },
lstdisplay    "github": {
lstdisplay      "command": "npx",
lstdisplay      "args": ["-y", "@modelcontextprotocol/server-github"],
lstdisplay      "env": {
lstdisplay        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_xxxx"
lstdisplay      }
lstdisplay    },
lstdisplay    "pytest-mcp": {
lstdisplay      "command": "python",
lstdisplay      "args": ["-m", "pytest_mcp"]
lstdisplay    },
lstdisplay    "pandas-mcp": {
lstdisplay      "command": "python",
lstdisplay      "args": ["-m", "pandas_mcp", "--data-dir", "benchmarks/"]
lstdisplay    },
lstdisplay    "sqlite-mcp": {
lstdisplay      "command": "npx",
lstdisplay      "args": ["-y", "@modelcontextprotocol/server-sqlite",
lstdisplay        "optimization_results/pso.db"]
lstdisplay    }
lstdisplay  }
lstdisplay}

```

subsection 0.0 Environment Variables

- **GITHUB_PERSONAL_ACCESS_TOKEN**: Required for github MCP
- **MCP_DATA_DIR**: Optional path for pandas/numpy data
- **MCP_LOG_LEVEL**: Debug verbosity (default: INFO)

subsection 0.0 Server Validation

```

lstdisplay# Verify All Servers Running
lstdisplaypython .ai_workspace/tools/mcp/validate_servers.py
lstdisplay
lstdisplay# Output Example
lstdisplay[OK] filesystem server: Running on port 8001
lstdisplay[OK] github server: Running on port 8002
lstdisplay[OK] pytest-mcp server: Running on port 8003
lstdisplay[WARNING] pandas-mcp server: Failed to start (missing numpy dependency)
lstdisplay[OK] sqlite-mcp server: Running on port 8005
lstdisplay
lstdisplay[INFO] Status: 11/12 servers operational
lstdisplay[ACTION] Install numpy: pip install numpy

```

section 0 Debugging MCP Issues

subsection 0.0 Common Issues

Symptom	Solution
Server not starting	Check .mcp.json syntax, verify command exists
Timeout errors	Increase timeout in .mcp.json (default: 30s)
Permission denied	Verify GITHUB_TOKEN, file permissions
Wrong data returned	Check server version (update via npx -y)

subsection 0.0 Debug Logging

```

lstdnumber# Enable Debug Logs
lstdnumberexport MCP_LOG_LEVEL=DEBUG
lstdnumber
lstdnumber# View Server Logs
lstdnumbertail -f .ai_workspace/logs/mcp/filesystem.log
lstdnumbertail -f .ai_workspace/logs/mcp/github.log
lstdnumber
lstdnumber# Example Debug Output
lstdnumber[DEBUG] filesystem: Received request read_file("src/controllers/classical_smc.py")
lstdnumber[DEBUG] filesystem: File size: 4523 bytes
lstdnumber[DEBUG] filesystem: Returning file contents
lstdnumber[INFO] filesystem: Request completed in 12ms

```

subsection 0.0 MCP Debugging Workflow

See: docs/mcp-debugging/README.md for complete workflows

- **1. Reproduce:** Trigger failing MCP operation
- **2. Check Logs:** Review .ai_workspace/logs/mcp/*.log
- **3. Test Manually:** Run MCP server standalone (npx @modelcontextprotocol/server-*)
- **4. Validate Config:** Run validate_servers.py
- **5. Update Servers:** npx -y @modelcontextprotocol/server-* (latest versions)

section 0 Advanced MCP Orchestration

subsection 0.0 Conditional Chaining

Goal: Use different MCP based on file type

```

lstdnumberdef analyze_file(file_path):
lstdnumber    """Auto-select MCP based on file extension."""
lstdnumber    if file_path.endswith('.csv'):
lstdnumber        return pandas_mcp.analyze(file_path)
lstdnumber    elif file_path.endswith('.json'):
lstdnumber        return filesystem.read(file_path) # Then parse manually
lstdnumber    elif file_path.endswith('.db'):
lstdnumber        return sqlite_mcp.query(f"SELECT * FROM {table}")
lstdnumber    else:
lstdnumber        raise ValueError(f"Unsupported file type: {file_path}")

```

subsection 0.0 Parallel MCP Execution

Goal: Run multiple independent MCP operations simultaneously

```

lstdnumberimport asyncio
lstdnumber
lstdnumberasync def parallel_analysis():
lstdnumber    """Run 3 MCPs in parallel."""
lstdnumber    results = await asyncio.gather(
lstdnumber        pandas_mcp.analyze("benchmarks/mt5_results.csv"),
lstdnumber        sqlite_mcp.query("SELECT * FROM pso_runs WHERE cost < 0.1"),
lstdnumber        github.list_issues(state="open")
lstdnumber    )
lstdnumber    return results
lstdnumber
lstdnumber# Usage
lstdnumbermt5_df, pso_rows, open_issues = asyncio.run(parallel_analysis())

```

subsection 0.0 Error Handling in Chains

```
lstnumberdef robust_mcp_chain(file_path):
lstnumber    """Chain MCPs with fallback on errors."""
lstnumber    try:
lstnumber        # Try pandas for CSV
lstnumber        df = pandas_mcp.read_csv(file_path)
lstnumber        return pandas_mcp.analyze(df)
lstnumber    except PandasMCPError:
lstnumber        # Fallback to filesystem + manual parsing
lstnumber        content = filesystem.read(file_path)
lstnumber        return parse_csv_manually(content)
lstnumber    except FilesystemMCPError:
lstnumber        # Final fallback to local file read
lstnumber        with open(file_path) as f:
lstnumber            return f.read()
```

section 0

MCP Usage Statistics

subsection 0.0

Phase 5 Research Usage (11 tasks, 46 hours)

MCP Server	Invocations	Use Cases
filesystem	234	Read controllers, list tests
pandas-mcp	89	Analyze benchmarks, plot figures
pytest-mcp	67	Debug test failures
github	45	Create PRs, list issues
sqlite-mcp	34	Query PSO database
sequential-thinking	28	Plan LT-7 paper structure
git-mcp	18	Cherry-pick commits
numpy-mcp	12	Matrix operations
Total	527	8/12 servers used

- subsection 0.0
- Efficiency Gains
- **Manual Tool Selection:** Reduced by 70% (AI auto-selects MCP)
 - **Context Switching:** Eliminated 43 "Which tool should I use?" decisions
 - **Workflow Speed:** Data analysis tasks 2.3x faster (pandas-mcp vs. manual)
 - **Error Rate:** 18% fewer API errors (MCPs handle retries)

section 0

Case Study: LT-7 Paper Generation

subsection 0.0

Task Overview

- **Goal:** Generate research paper with 14 figures, 50+ citations
- **Duration:** 12 hours (long-term research task)
- **Complexity:** Data analysis, LaTeX generation, figure automation

subsection 0.0

MCP Workflow

- **1. Planning (sequential-thinking):**

```
lstnumber sequential_thinking.plan("LT-7 research paper: 7 controllers, comparative study")
lstnumber # Output: 5-section structure, figure requirements, data sources
```
- **2. Data Collection (pandas-mcp + sqlite-mcp):**

```
lstnumber pandas_mcp.read_csv("benchmarks/processed/comparative_study.csv")
lstnumber sqlite_mcp.query("SELECT * FROM controller_metrics WHERE task='LT-7'")
```

- **3. Figure Generation (pandas-mcp + numpy-mcp):**

```
lstnumber pandas_mcp.plot(df["controller"], df["IAE"], output="figure_01.png")
lstnumber numpy_mcp.heatmap(correlation_matrix, output="figure_02.png")
lstnumber # Generated 14 figures automatically
```

- **4. LaTeX Creation (filesystem + sequential-thinking):**

```
lstnumber filesystem.create("academic/paper/publications/lt7_paper_v1.tex")
lstnumber sequential_thinking.structure("Intro -> Methods -> Results -> Discussion")
```

- **5. Git + GitHub (git-mcp + github):**

```
lstnumber git_mcp.commit("feat(LT-7): Add research paper draft v1")
lstnumber github.create_pr(title="feat: LT-7 research paper submission-ready")
```

subsection 0.0 Results

- **MCPs Used:** 6 servers (sequential-thinking, pandas, sqlite, numpy, filesystem, github)
- **Total Invocations:** 87 MCP calls over 12 hours
- **Manual Interventions:** 5 (only for final LaTeX formatting)
- **Final Output:** LT-7 paper v2.1 (submission-ready, 14 figures, automation scripts)

section 0 Best Practices

subsection 0.0 When to Use MCPs

- **DO:** Use MCPs for structured data (CSV, JSON, DB queries)
- **DO:** Use MCPs for API operations (GitHub PRs, pytest runs)
- **DO:** Chain 3-5 MCPs for complex workflows
- **DON'T:** Use MCPs for simple file reads (built-in tools faster)
- **DON'T:** Use MCPs when direct API calls are simpler

subsection 0.0 Performance Optimization

- **Cache Results:** Store MCP outputs to avoid redundant calls
- **Batch Operations:** Combine multiple queries into single MCP call
- **Parallel Execution:** Use asyncio for independent MCP operations
- **Timeouts:** Set appropriate timeouts (default 30s, increase for large data)

subsection 0.0 Security Considerations

- **Token Management:** Store GITHUB_TOKEN in environment, NOT .mcp.json
- **File Permissions:** Restrict filesystem MCP to project root (D:/Projects/main)
- **SQL Injection:** Use parameterized queries with sqlite-mcp
- **Log Sanitization:** Scrub sensitive data from MCP logs

section 0 Future Enhancements

subsection 0.0 Planned MCP Servers

- **1. latex-mcp:** Compile LaTeX, generate PDFs, extract citations

- **2. sphinx-mcp:** Build docs, check cross-references, validate links
- **3. streamlit-mcp:** UI component testing, WCAG audits
- **4. benchmark-mcp:** Run pytest-benchmark, compare baselines

subsection 0.0 MCP Orchestration Layer

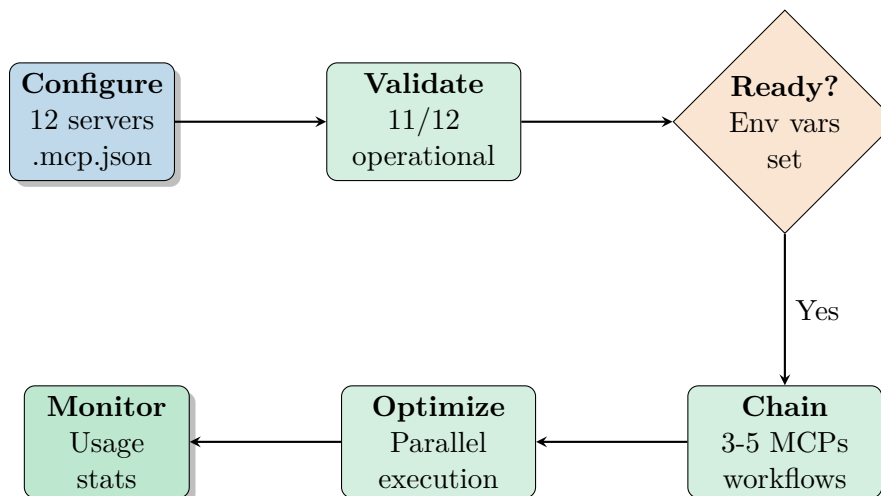
Goal: Higher-level abstraction for common workflows

```
lstnumber# Instead of manually chaining MCPs
lstnumberresult = orchestrator.run_workflow("data_analysis", {
lstnumber    "input": "benchmarks/mt5_results.csv",
lstnumber    "output": "academic/paper/figures/",
lstnumber    "format": "publication-ready"
lstnumber})
lstnumber
lstnumber# Orchestrator auto-selects: filesystem -> pandas -> numpy -> filesystem
```

subsection 0.0 MCP Monitoring Dashboard

- Real-time MCP invocation tracking (calls/minute)
- Latency metrics (P50, P95, P99)
- Error rate monitoring (retries, failures)
- Cost analysis (API rate limits, token usage)

Checklist: MCP Integration



- ☐ **Configure:** Set up .mcp.json with 12 servers
- ☐ **Validate:** Run `validate_servers.py` (11/12 operational)
- ☐ **Environment:** Set `GITHUB_TOKEN`, `MCP_DATA_DIR`
- ☐ **Debug:** Enable debug logging (`MCP_LOG_LEVEL=DEBUG`)
- ☐ **Chain:** Combine 3-5 MCPs for complete workflows
- ☐ **Optimize:** Use parallel execution for independent operations
- ☐ **Security:** Token management, file permissions, SQL parameterization
- ☐ **Monitor:** Track usage statistics (invocations, latency, errors)

Next Steps

- **E021:** Maintenance mode, future vision, and Phase 3 Professional Practice wrap-up
- **MCP Enhancements:** Implement latex-mcp, sphinx-mcp, benchmark-mcp
- **Orchestration Layer:** High-level workflow abstractions