

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)

**enumfilesystem:** File operations (read, write, list, search)

0. **enumgithub:** Issues, PRs, checks, releases
0. **enumsequential-thinking:** Planning, debugging, reasoning
0. **enumipuppeteer:** UI testing, browser automation

### subsection 0.0 Development Servers (4)

0. **enumpytest-mcp:** Test debugging, coverage analysis
0. **enumgit-mcp:** Advanced Git operations (rebase, cherry-pick, blame)
0. **enumimcp-analyzer:** Code quality, linting, static analysis
0. **enumimcp-debugger:** API testing, HTTP requests

### subsection 0.0 Data Science Servers (3)

0. **enumisqlite-mcp:** PSO optimization database queries
0. **enumipandas-mcp:** Data analysis, CSV/JSON processing
0. **enuminumpy-mcp:** Numerical compute, matrix operations

### subsection 0.0 Audit Server (1)

0. **enumilighthouse-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_smcl failing?"
lstnumber# AI auto-triggers:
    pytest_mcp.debug("tests/test_controllers/test_classical_smcl.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_smcl::test_convergence")
lstnumber3. puppeteer.screenshot("UI elements for manual inspection")
lstnumber4. mcp_analyzer.lint("src/controllers/classical_smcl.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

```

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

```

## subsection 0.0 Environment Variables

0. **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

```

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

```

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

```

## subsection 0.0 MCP Debugging Workflow

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

**enumiReproduce:** Trigger failing MCP operation

0. **enumiCheck Logs:** Review .ai\_workspace/logs/mcp/\*.log
0. **enumiTest Manually:** Run MCP server standalone (`npx @modelcontextprotocol/server-*`)
0. **enumiValidate Config:** Run `validate_servers.py`
0. **enumiUpdate 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

```

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

```

### subsection 0.0 Parallel MCP Execution

**Goal:** Run multiple independent MCP operations simultaneously

```

lstnumberimport asyncio
lstnumber
lstnumberasync def parallel_analysis():
lstnumber    """Run 3 MCPs in parallel."""
lstnumber    results = await asyncio.gather(
lstnumber        pandas_mcp.analyze("benchmarks/mt5_results.csv"),
lstnumber        sqlite_mcp.query("SELECT * FROM pso_runs WHERE cost < 0.1"),
lstnumber        github.list_issues(state="open")
lstnumber    )
lstnumber    return results
lstnumber
lstnumber# Usage
lstnumbermt5_df, pso_rows, open_issues = asyncio.run(parallel_analysis())

```

### subsection 0.0 Error Handling in Chains

```

lstnumber def 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
0. github	234	Read controllers, list tests
	89	Analyze benchmarks, plot figures
	67	Debug test failures
	45	Create PRs, list issues
	34	Query PSO database
	28	Plan LT-7 paper structure
	18	Cherry-pick commits
	12	Matrix operations
	<b>527</b>	<b>8/12 servers used</b>

### 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

enumiPlanning (sequential-thinking):

```

lstnumber sequential_thinking.plan("LT-7 research paper: 7 controllers, comparative
lstnumber     study")
lstnumber # Output: 5-section structure, figure requirements, data sources

```

0. enumiData 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'")

```

## 0. enumiFigure 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
```

## 0. enumiLaTeX Creation (filesystem + sequential-thinking):

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

## 0. enumiGit + 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

**0. 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

**enumilatex-mcp:** Compile LaTeX, generate PDFs, extract citations

0. enumisphinx-mcp: Build docs, check cross-references, validate links
0. enumistreamlit-mcp: UI component testing, WCAG audits
0. enumibenchmark-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

0. 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